2018

ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

ΕΡΓΑΣΙΑ 2018-2019

Π16036	ΙΩΑΝΝΙΔΗΣ ΠΑΝΑΓΙΩΤΗΣ
Π16097	ΝΙΚΑΣ ΔΙΟΝΥΣΗΣ
Π16112	ΠΑΡΑΒΑΝΤΗΣ ΑΘΑΝΑΣΙΟΣ

Περιεχόμενα

1 Μέρος 1ο

1.1 Εισαγωγή

Το 1ο ερώτημα της εργασίας προϋποθέτει τη δημιουργία μιας βάσης δεδομένων που θα περιέχει όλα τα στοιχεία των δυο CSV αρχείων που μας δίνονται. Για την σωστή ακολουθία ενεργειών, έχουμε συμπεριλάβει το αρχείο **create_table.sql** που βρίσκεται στο φάκελο **part_1**.

Το αρχείο είναι χωρισμένο σε κατηγορίες που περιέχον τις εντολές για τις παρακάτω ενέργειες:

- Δημιουργία του πίνακα **accident_information**.
- Αντιγραφή των δεδομένων του αρχείου **db2_Accident_Information.csv** στον πίνακα **accident_information**.
- Δημιουργία του πίνακα **vehicle_information**.
- Αντιγραφή των δεδομένων του αρχείου **db2_Vehicle_Information.csv** στον πίνακα **vehicle_information**.

Τα σχετικά COPY queries δεν περιλαμβάνουν έγκυρα path για τα .csv αρχεία, επομένως είναι αναγκαία η αντικατάστασή τους με τα σωστά path για το εκάστοτε προγραμματιστικό περιβάλλον.

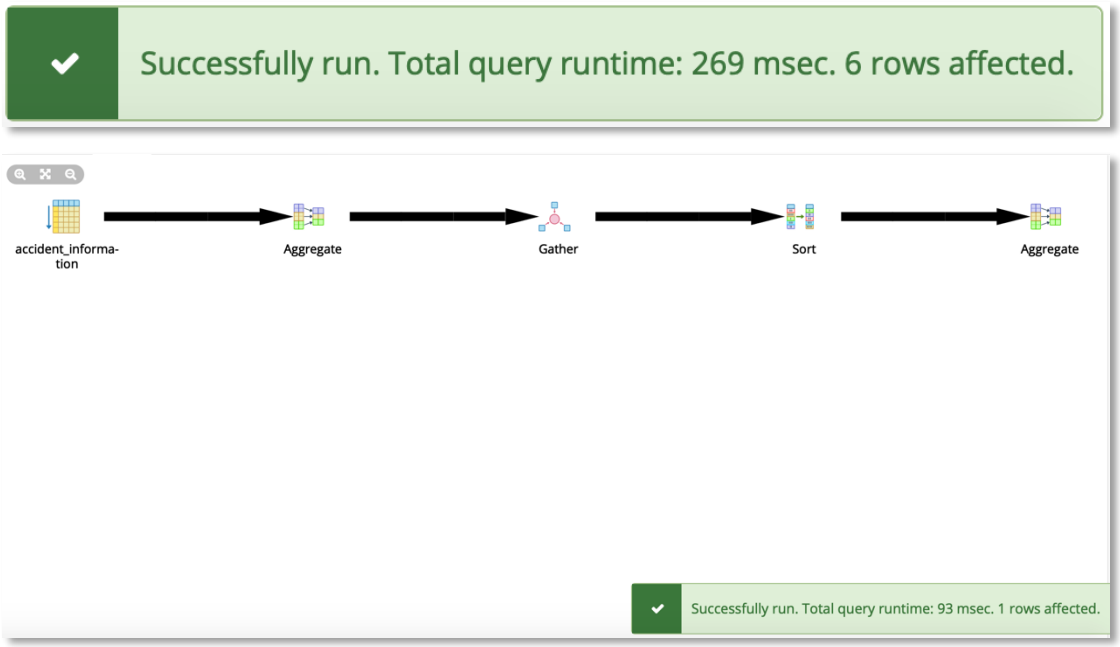
Οι απαντήσεις για κάθε ερώτημα του 1ου μέρους βρίσκονται στους φακέλους A_queries, B_shared_buffers, C_parallel_query, D_indexes και E_partitions αντίστοιχα, οι οποίοι είναι μέσα στο φάκελο **part_1**.

1.2 Ερώτημα Α

Στο πρώτο ερώτημα δημιουργούνται τα queries που θα χρησιμοποιηθούν κατά τη διάρκεια του πρώτου μέρους της εργασίας. Πριν την εκτέλεσή τους εκτελούμε την εντολή VACUMM FULL όπως ζητήθηκε για να κάνουν refresh τα στατιστικά των πινάκων. Στη συνέχεια θα εκτελέσουμε ένα με τη σειρά τα queries χρησιμοποιώντας τις προεπιλεγμένες ρυθμίσεις της Postgres και χωρίς να έχουμε δημιουργήσει ευρετήρια.

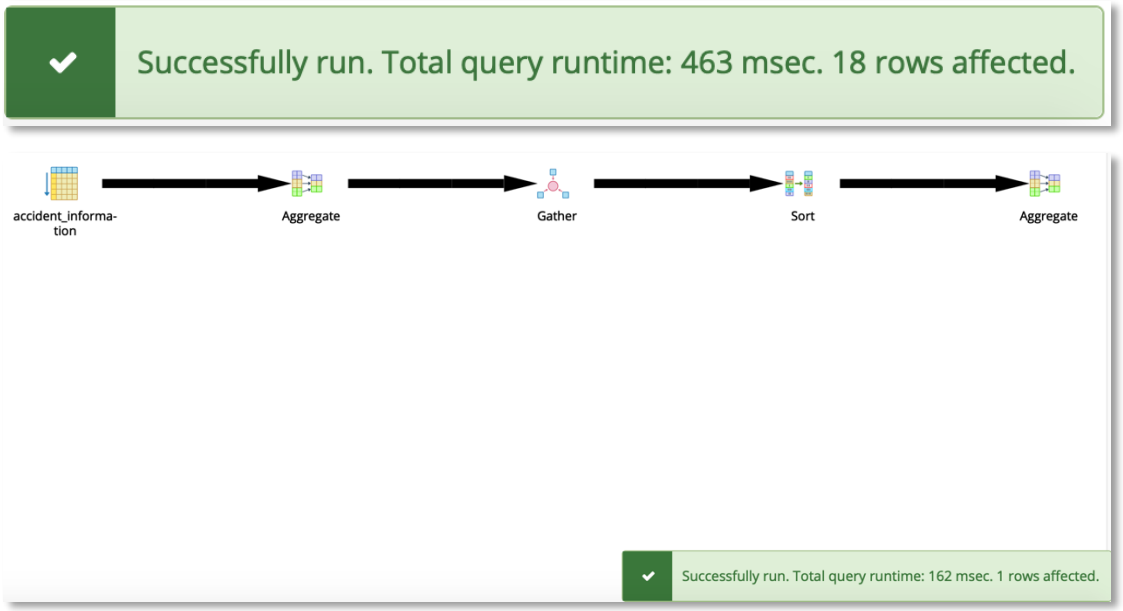
Έτσι έχουμε για κάθε query πρώτα το χρόνο εκτέλεσης, μετά την ανάλυση με EXPLAIN και τέλος το QUERY PLAN :

i) Πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου (road class)



	QUERY PLAN text
1	Finalize GroupAggregate (cost=37052.44..37052.59 rows=6 width=13)
2	Group Key: first_road_class
3	-> Sort (cost=37052.44..37052.47 rows=12 width=13)
4	Sort Key: first_road_class
5	-> Gather (cost=37050.96..37052.22 rows=12 width=13)
6	Workers Planned: 2
7	-> Partial HashAggregate (cost=36050.96..36051.02 rows=6 width=13)
8	Group Key: first_road_class
9	-> Parallel Seq Scan on accident_information (cost=0.00..32056.64 rows=79...

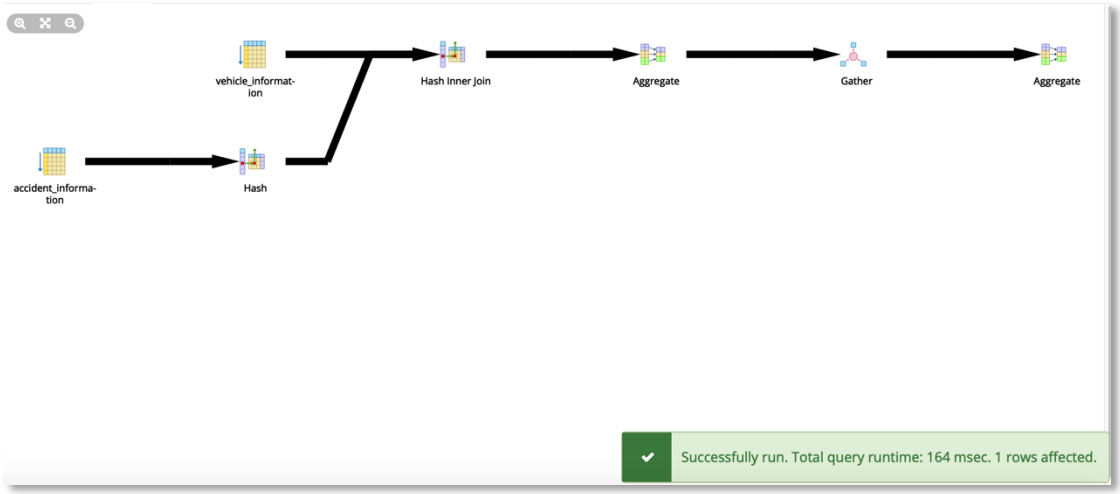
ii) Πόσα είναι τα ατυχήματα ανά κατηγορία δρόμου και ανά κατηγορία ατυχήματος (Accident Severity).



	QUERY PLAN text
1	Finalize GroupAggregate (cost=39052.83..39053.37 rows=18 width=20)
2	Group Key: first_road_class, accident_severity
3	-> Sort (cost=39052.83..39052.92 rows=36 width=20)
4	Sort Key: first_road_class, accident_severity
5	-> Gather (cost=39048.12..39051.90 rows=36 width=20)
6	Workers Planned: 2
7	-> Partial HashAggregate (cost=38048.12..38048.30 rows=18 width=20)
8	Group Key: first_road_class, accident_severity
9	-> Parallel Seq Scan on accident_information (cost=0.00..32056.64 rows=79...

iii) Πόσα ατυχήματα έγιναν σε αστική περιοχή πριν από την 1/1/2010 και η ηλικιακή κατηγορία του οδηγού είναι 26-35.

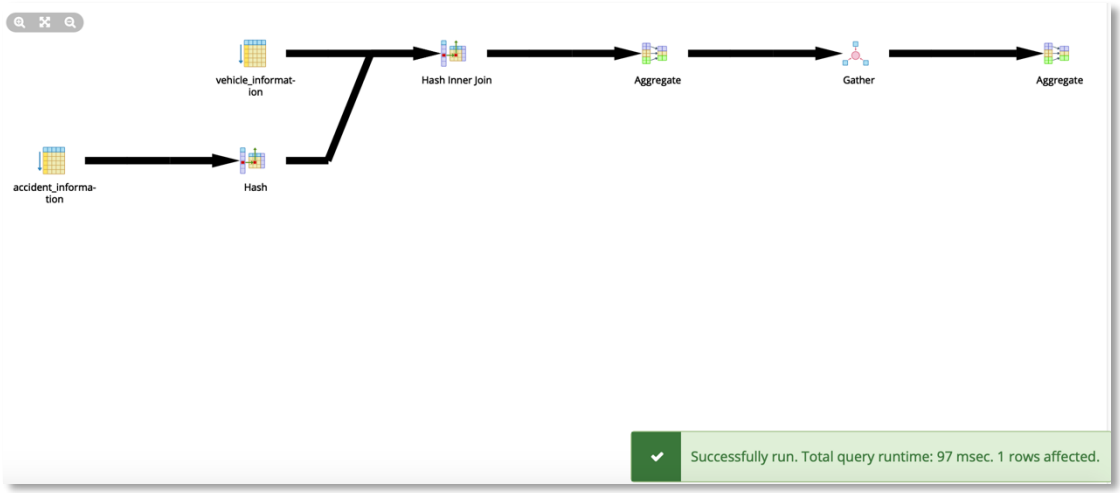
Successfully run. Total query runtime: 1 secs. 1 rows affected.



	QUERY PLAN text
1	Finalize GroupAggregate (cost=64638.46..114466.00 rows=36 width=23)
2	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area
3	-> Gather (cost=64638.46..114465.10 rows=72 width=23)
4	Workers Planned: 2
5	-> Partial GroupAggregate (cost=63638.46..113457.90 rows=36 width=23)
6	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area
7	-> Hash Join (cost=63638.46..113028.28 rows=57235 width=108)
8	Hash Cond: ((vehicles.accident_index)::text = (accidents.accident_index)::text)
9	-> Parallel Seq Scan on vehicle_information vehicles (cost=0.00..38115.61 rows=186362 wi...
10	Filter: ((age_band_of_driver)::text = '26 - 35'::text)
11	-> Hash (cost=52827.11..52827.11 rows=588828 width=20)
12	-> Seq Scan on accident_information accidents (cost=0.00..52827.11 rows=588828 widt...
13	Filter: ((date < '2010-01-01'::date) AND ((urban_or_rural_area)::text = 'Urban'::text))

iv) Πόσα ατυχήματα έγιναν σε αγροτική περιοχή το έτος 2012 και η ηλικιακή κατηγορία του οδηγού είναι 36-45.

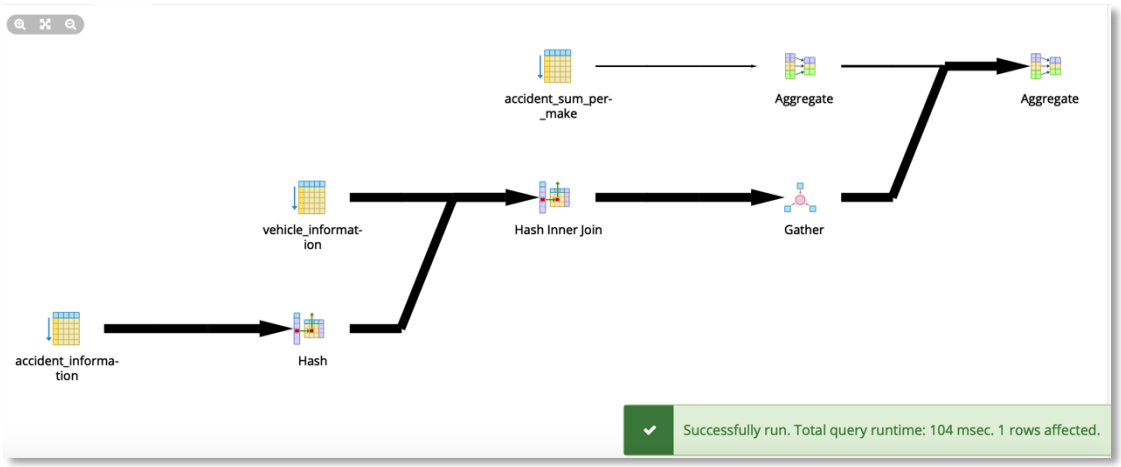
Successfully run. Total query runtime: 683 msec. 1 rows affected.



	QUERY PLAN text
1	Finalize GroupAggregate (cost=54461.76..93461.28 rows=432 width=27)
2	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area, accid...
3	-> Gather (cost=54461.76..93448.32 rows=864 width=27)
4	Workers Planned: 2
5	-> Partial GroupAggregate (cost=53461.76..92361.92 rows=432 width=27)
6	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_are...
7	-> Hash Join (cost=53461.76..92309.32 rows=4828 width=112)
8	Hash Cond: ((vehicles.accident_index)::text = (accidents.accident_in...
9	-> Parallel Seq Scan on vehicle_information vehicles (cost=0.00..38...
10	Filter: ((age_band_of_driver)::text = '36 - 45'::text)
11	-> Hash (cost=52827.11..52827.11 rows=50772 width=24)
12	-> Seq Scan on accident_information accidents (cost=0.00..528...
13	Filter: (((urban_or_rural_area)::text = 'Rural'::text) AND (year =...

v) Ποιος είναι ο κατασκευαστής του οχήματος που έχει τα περισσότερα ατυχήματα μεταξύ των ετών 2010 και 2012, η ηλικιακή κατηγορία των οδηγών είναι 26- 35 και η κατηγορία δρόμου είναι Α.

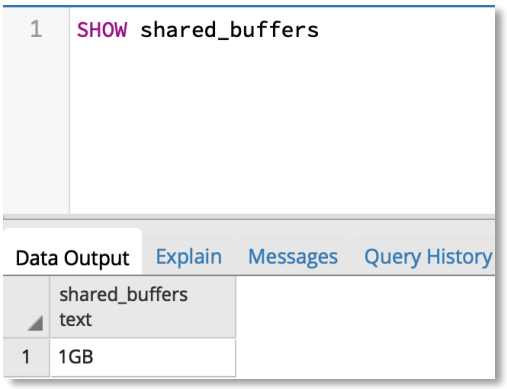
Successfully run. Total query runtime: 781 msec. 1 rows affected.



	QUERY PLAN text
1	HashAggregate (cost=110439.30..110609.94 rows=17064 width=29)
2	Group Key: v.make, v.age_band_of_driver, a.first_road_class
3	Filter: (count(v.make) = $0)
4	InitPlan 1 (returns $0)
5	-> Aggregate (cost=4.71..4.72 rows=1 width=8)
6	-> Seq Scan on accident_sum_per_make (cost=0.00..4.17 rows=217 width=8)
7	-> Gather (cost=62351.11..109842.07 rows=47400 width=21)
8	Workers Planned: 2
9	-> Hash Join (cost=61351.11..104102.07 rows=19750 width=21)
10	Hash Cond: ((v.accident_index)::text = (a.accident_index)::text)
11	-> Parallel Seq Scan on vehicle_information v (cost=0.00..38115.61 rows=186362 width=30)
12	Filter: ((age_band_of_driver)::text = '26 - 35'::text)
13	-> Hash (cost=57620.30..57620.30 rows=203185 width=19)
14	-> Seq Scan on accident_information a (cost=0.00..57620.30 rows=203185 width=19)
15	Filter: ((year >= 2010) AND (year <= 2012) AND ((first_road_class)::text = 'A'::text))

1.2 Ερώτημα Β

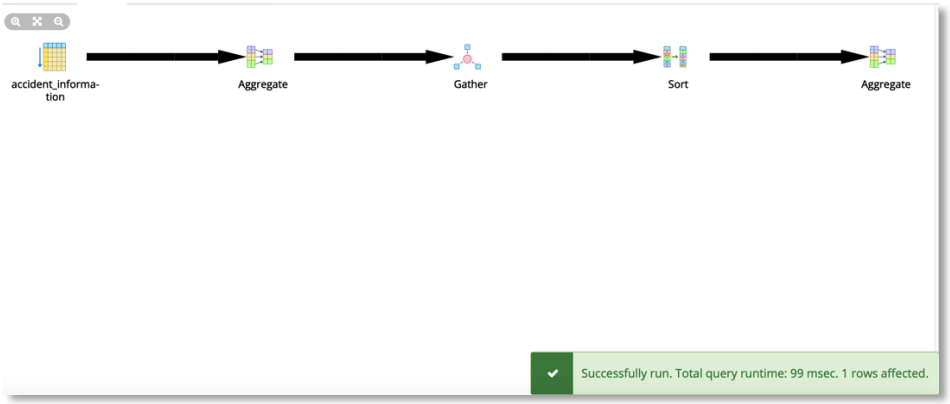
Στο δεύτερο ερώτημα θα ξανά εκτελέσουμε τα queries του προηγούμενου ερωτήματος αλλά αυτή τη φορά θα χρησιμοποιήσουμε ως buffer περισσότερη μνήμη από τη μνήμη RAM του υπολογιστή μας. Πιο συγκεκριμένα από τα 128MB που είναι το default της postgres θα το αυξήσουμε στο 1024MB. Έτσι μετά τη χρήση της εντολής **ALTER SYSTEM SET shared_buffers TO '1024'MB** και επανεκκίνηση της postgres θα έχουμε το εξής αποτέλεσμα.



Έτσι έχουμε για κάθε query πρώτα το χρόνο εκτέλεσης, μετά την ανάλυση με EXPLAIN και τέλος το QUERY PLAN :

i) Πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου (road class)

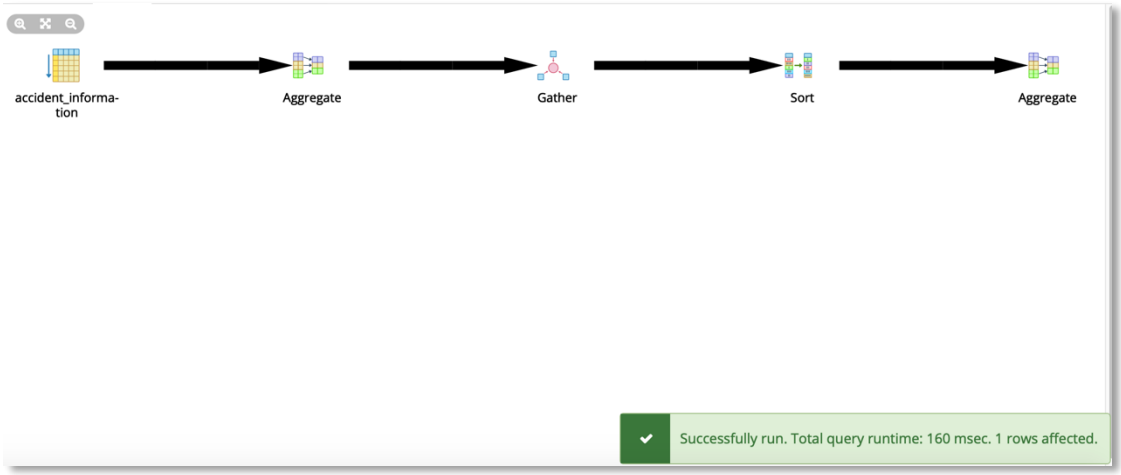
Successfully run. Total query runtime: 256 msec. 6 rows affected.



	QUERY PLAN text
1	Finalize GroupAggregate (cost=37052.44..37052.59 rows=6 width=13)
2	Group Key: first_road_class
3	-> Sort (cost=37052.44..37052.47 rows=12 width=13)
4	Sort Key: first_road_class
5	-> Gather (cost=37050.96..37052.22 rows=12 width=13)
6	Workers Planned: 2
7	-> Partial HashAggregate (cost=36050.96..36051.02 rows=6 width=13)
8	Group Key: first_road_class
9	-> Parallel Seq Scan on accident_information (cost=0.00..32056.64 rows=798864 width=5)

ii) Πόσα είναι τα ατυχήματα ανά κατηγορία δρόμου και ανά κατηγορία ατυχήματος (Accident Severity).

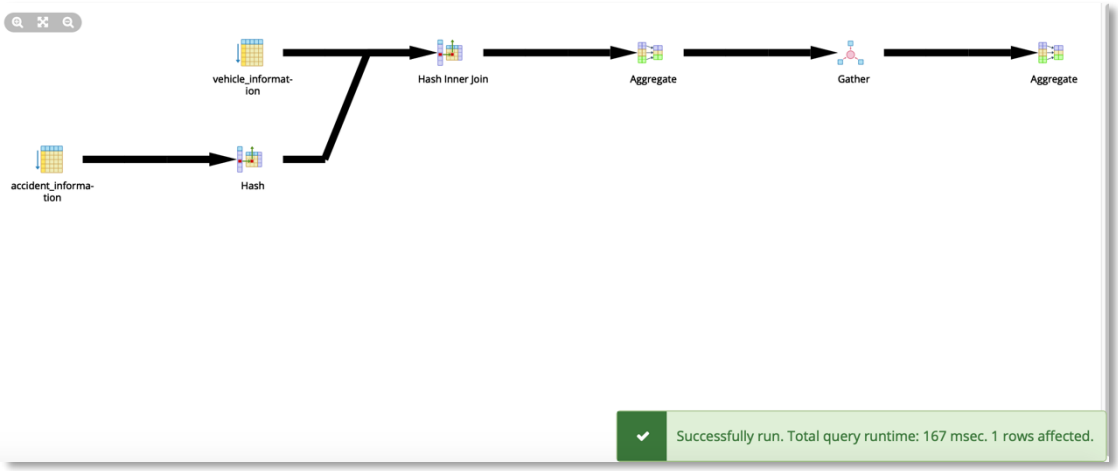
Successfully run. Total query runtime: 479 msec. 18 rows affected.



	QUERY PLAN text
1	Finalize GroupAggregate (cost=39052.83..39053.37 rows=18 width=20)
2	Group Key: first_road_class, accident_severity
3	-> Sort (cost=39052.83..39052.92 rows=36 width=20)
4	Sort Key: first_road_class, accident_severity
5	-> Gather (cost=39048.12..39051.90 rows=36 width=20)
6	Workers Planned: 2
7	-> Partial HashAggregate (cost=38048.12..38048.30 rows=18 width=20)
8	Group Key: first_road_class, accident_severity
9	-> Parallel Seq Scan on accident_information (cost=0.00..32056.64 rows=798864 width=101)

iii) Πόσα ατυχήματα έγιναν σε αστική περιοχή πριν από την 1/1/2010 και η ηλικιακή κατηγορία του οδηγού είναι 26-35.

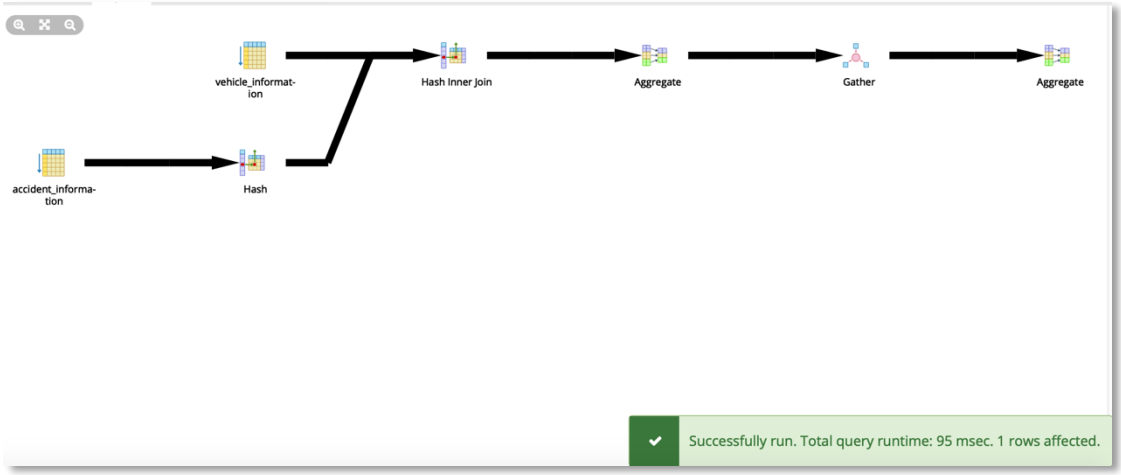
Successfully run. Total query runtime: 1 secs. 1 rows affected.



	QUERY PLAN text
1	Finalize GroupAggregate (cost=64638.46..114466.00 rows=36 width=23)
2	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area
3	-> Gather (cost=64638.46..114465.10 rows=72 width=23)
4	Workers Planned: 2
5	-> Partial GroupAggregate (cost=63638.46..113457.90 rows=36 width=23)
6	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area
7	-> Hash Join (cost=63638.46..113028.28 rows=57235 width=108)
8	Hash Cond: ((vehicles.accident_index)::text = (accidents.accident_index)::text)
9	-> Parallel Seq Scan on vehicle_information vehicles (cost=0.00..38115.61 rows=186362 width=116)
10	Filter: ((age_band_of_driver)::text = '26 - 35'::text)
11	-> Hash (cost=52827.11..52827.11 rows=588828 width=20)
12	-> Seq Scan on accident_information accidents (cost=0.00..52827.11 rows=588828 width=20)
13	Filter: ((date < '2010-01-01'::date) AND ((urban_or_rural_area)::text = 'Urban'::text))

iv) Πόσα ατυχήματα έγιναν σε αγροτική περιοχή το έτος 2012 και η ηλικιακή κατηγορία του οδηγού είναι 36-45.

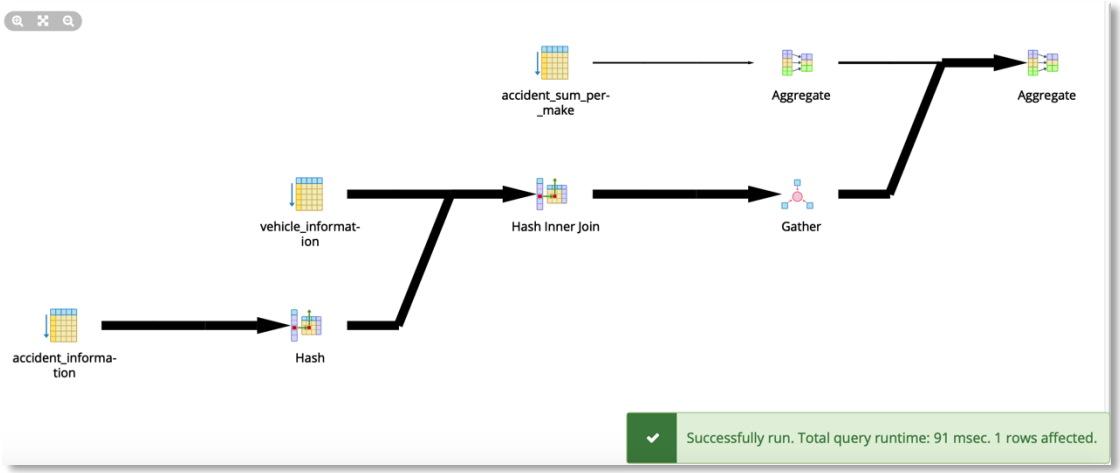
Successfully run. Total query runtime: 616 msec. 1 rows affected.



	QUERY PLAN text
1	Finalize GroupAggregate (cost=54461.76..93461.28 rows=432 width=27)
2	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area, accidents.year
3	-> Gather (cost=54461.76..93448.32 rows=864 width=27)
4	Workers Planned: 2
5	-> Partial GroupAggregate (cost=53461.76..92361.92 rows=432 width=27)
6	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area, accidents.year
7	-> Hash Join (cost=53461.76..92309.32 rows=4828 width=112)
8	Hash Cond: ((vehicles.accident_index)::text = (accidents.accident_index)::text)
9	-> Parallel Seq Scan on vehicle_information vehicles (cost=0.00..38115.61 rows=182311 width=116)
10	Filter: ((age_band_of_driver)::text = '36 - 45'::text)
11	-> Hash (cost=52827.11..52827.11 rows=50772 width=24)
12	-> Seq Scan on accident_information accidents (cost=0.00..52827.11 rows=50772 width=24)
13	Filter: (((urban_or_rural_area)::text = 'Rural'::text) AND (year = 2012))

v) Ποιος είναι ο κατασκευαστής του οχήματος που έχει τα περισσότερα ατυχήματα μεταξύ των ετών 2010 και 2012, η ηλικιακή κατηγορία των οδηγών είναι 26- 35 και η κατηγορία δρόμου είναι Α.

Successfully run. Total query runtime: 764 msec. 1 rows affected.



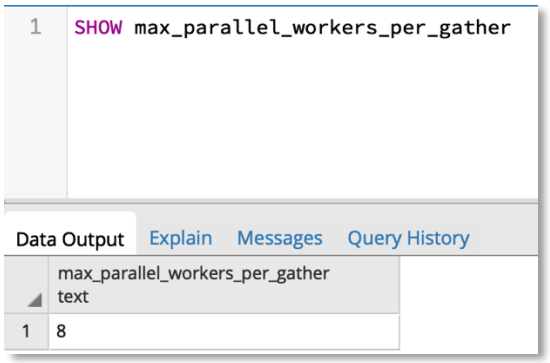
	QUERY PLAN text
1	HashAggregate (cost=110439.30..110609.94 rows=17064 width=29)
2	Group Key: v.make, v.age_band_of_driver, a.first_road_class
3	Filter: (count(v.make) = $0)
4	InitPlan 1 (returns $0)
5	-> Aggregate (cost=4.71..4.72 rows=1 width=8)
6	-> Seq Scan on accident_sum_per_make (cost=0.00..4.17 rows=217 width=8)
7	-> Gather (cost=62351.11..109842.07 rows=47400 width=21)
8	Workers Planned: 2
9	-> Hash Join (cost=61351.11..104102.07 rows=19750 width=21)
10	Hash Cond: ((v.accident_index)::text = (a.accident_index)::text)
11	-> Parallel Seq Scan on vehicle_information v (cost=0.00..38115.61 rows=186362 width=30)
12	Filter: ((age_band_of_driver)::text = '26 - 35'::text)
13	-> Hash (cost=57620.30..57620.30 rows=203185 width=19)
14	-> Seq Scan on accident_information a (cost=0.00..57620.30 rows=203185 width=19)
15	Filter: ((year >= 2010) AND (year <= 2012) AND ((first_road_class)::text = 'A'::text))

Συμπέρασμα

Όπως βλέπουμε υπάρχουν ελάχιστες βελτιώσεις στο χρόνο εκτέλεσης κάθε ερωτήματος και στα δύο τελευταία πολλές φορές ίσως και χειρότερο χρόνο εκτέλεσης. Άρα συμπεραίνουμε ότι ακόμα και αν τα shared_buffers μας έχουν 1GB ελεύθερο χώρο για να φορτώσουμε ακόμα και ολόκληρα τα dataset στη μνήμη RAM δεν υπάρχει κάποια αξιοσημείωτη βελτίωση.

1.3 Ερώτημα Γ

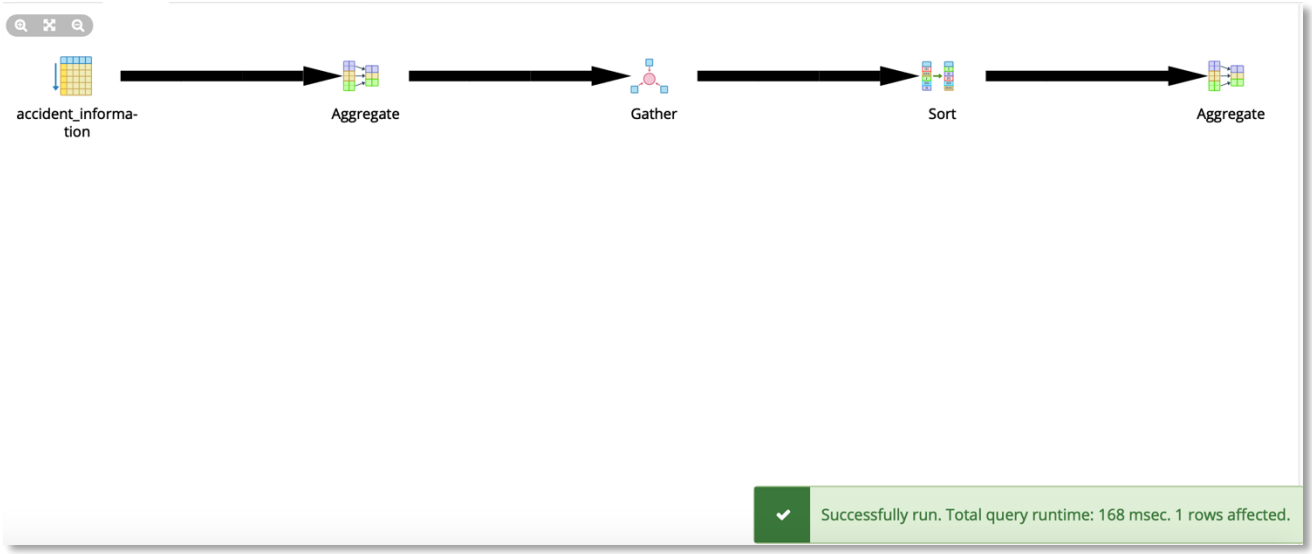
Στο τρίτο ερώτημα θα ξανά εκτελέσουμε τα queries του πρώτου ερωτήματος αλλά αυτή τη φορά θα χρησιμοποιήσουμε όλη την επεξεργαστική ισχύ του υπολογιστή μας. Πιο συγκεκριμένα από τους 2 parallel_workers_per_gather που είναι το default της postgres θα το αυξήσουμε στους 8. Έτσι μετά τη χρήση της εντολής **ALTER SYSTEM SET max_parallel_workers_per_gather TO 8** και επανεκκίνηση της postgres θα έχουμε το εξής αποτέλεσμα.



Έτσι έχουμε για κάθε query πρώτα το χρόνο εκτέλεσης, μετά την ανάλυση με EXPLAIN και τέλος το QUERY PLAN :

i) Πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου (road class)

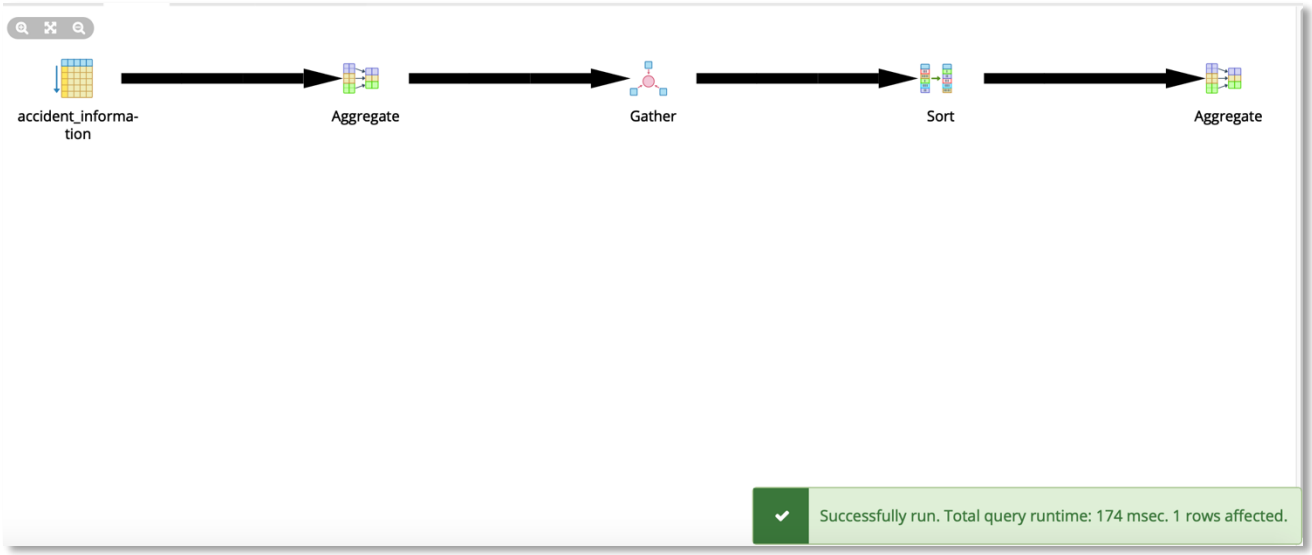
Successfully run. Total query runtime: 232 msec. 6 rows affected.



	QUERY PLAN text
1	Finalize GroupAggregate (cost=34347.37..34347.56 rows=6 width=13)
2	Group Key: first_road_class
3	-> Sort (cost=34347.37..34347.41 rows=18 width=13)
4	Sort Key: first_road_class
5	-> Gather (cost=34345.13..34346.99 rows=18 width=13)
6	Workers Planned: 3
7	-> Partial HashAggregate (cost=33345.13..33345.19 rows=6 width=13)
8	Group Key: first_road_class
9	-> Parallel Seq Scan on accident_information (cost=0.00..30252.75 rows=618475 width=5)

ii) Πόσα είναι τα ατυχήματα ανά κατηγορία δρόμου και ανά κατηγορία ατυχήματος (Accident Severity).

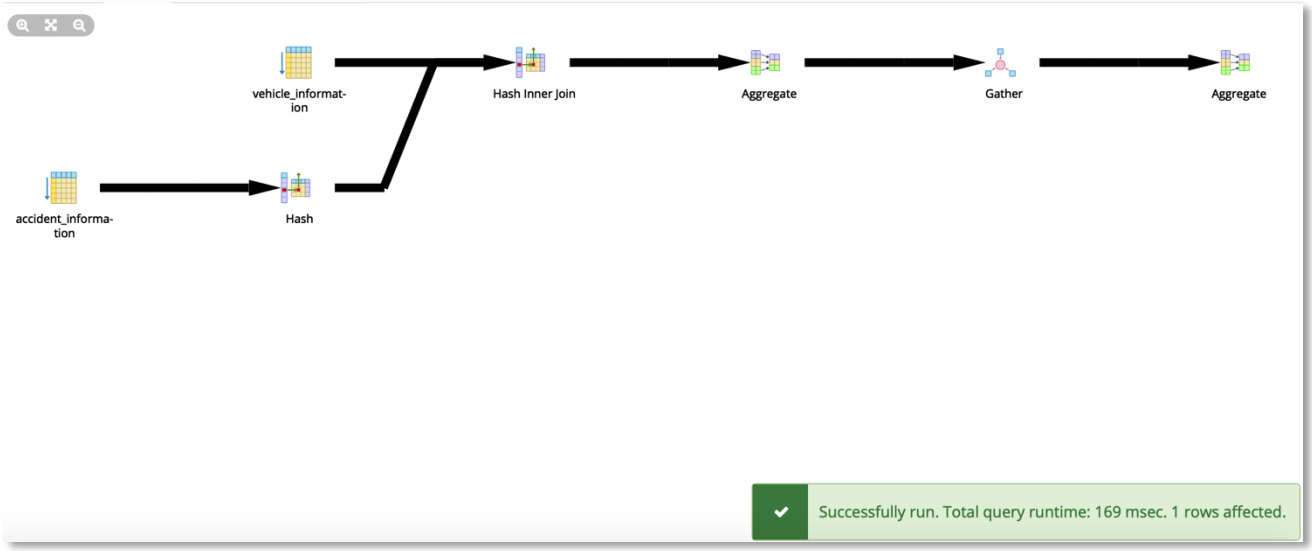
Successfully run. Total query runtime: 437 msec. 18 rows affected.



	QUERY PLAN text
1	Finalize GroupAggregate (cost=35898.45..35899.17 rows=18 width=20)
2	Group Key: first_road_class, accident_severity
3	-> Sort (cost=35898.45..35898.59 rows=54 width=20)
4	Sort Key: first_road_class, accident_severity
5	-> Gather (cost=35891.32..35896.90 rows=54 width=20)
6	Workers Planned: 3
7	-> Partial HashAggregate (cost=34891.32..34891.50 rows=18 width=20)
8	Group Key: first_road_class, accident_severity
9	-> Parallel Seq Scan on accident_information (cost=0.00..30252.75 rows=618475 width=101)

iii) Πόσα ατυχήματα έγιναν σε αστική περιοχή πριν από την 1/1/2010 και η ηλικιακή κατηγορία του οδηγού είναι 26-35.

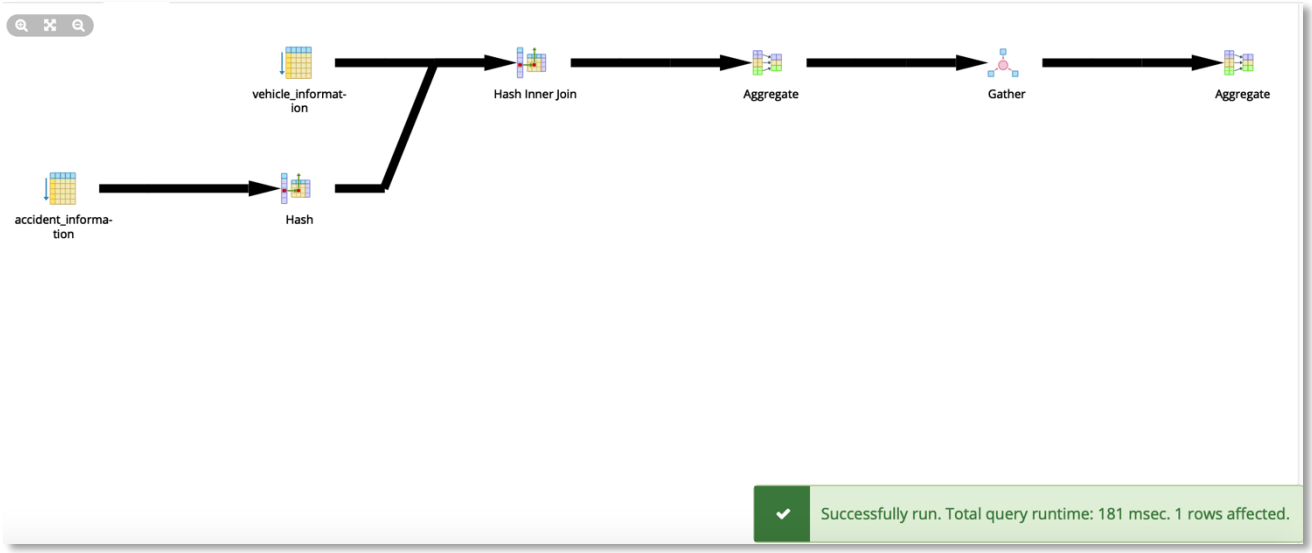
Successfully run. Total query runtime: 1 secs. 1 rows affected.



	QUERY PLAN text
1	Finalize GroupAggregate (cost=64638.46..110047.34 rows=36 width=23)
2	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area
3	-> Gather (cost=64638.46..110046.17 rows=108 width=23)
4	Workers Planned: 3
5	-> Partial GroupAggregate (cost=63638.46..109035.37 rows=36 width=23)
6	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area
7	-> Hash Join (cost=63638.46..108702.68 rows=44311 width=108)
8	Hash Cond: ((vehicles.accident_index)::text = (accidents.accident_index)::text)
9	-> Parallel Seq Scan on vehicle_information vehicles (cost=0.00..35555.05 rows=144281 width=116)
10	Filter: ((age_band_of_driver)::text = '26 - 35'::text)
11	-> Hash (cost=52827.11..52827.11 rows=588828 width=20)
12	-> Seq Scan on accident_information accidents (cost=0.00..52827.11 rows=588828 width=20)
13	Filter: ((date < '2010-01-01'::date) AND ((urban_or_rural_area)::text = 'Urban'::text))

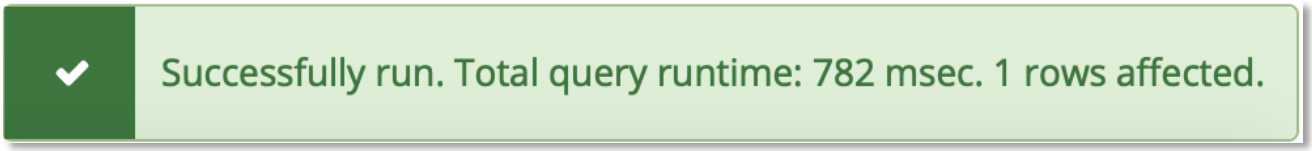
iv) Πόσα ατυχήματα έγιναν σε αγροτική περιοχή το έτος 2012 και η ηλικιακή κατηγορία του οδηγού είναι 36-45.

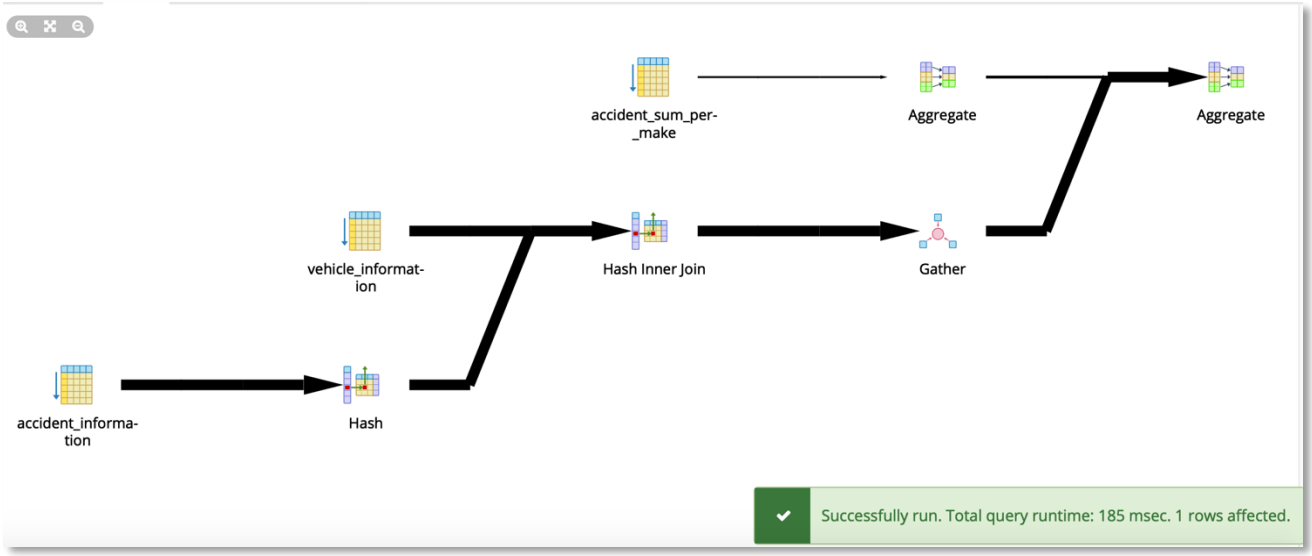
Successfully run. Total query runtime: 698 msec. 1 rows affected.



	QUERY PLAN text
1	Finalize GroupAggregate (cost=54461.76..90772.06 rows=432 width=27)
2	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area, accidents.year
3	-> Gather (cost=54461.76..90754.78 rows=1296 width=27)
4	Workers Planned: 3
5	-> Partial GroupAggregate (cost=53461.76..89625.18 rows=432 width=27)
6	Group Key: vehicles.age_band_of_driver, accidents.urban_or_rural_area, accidents.year
7	-> Hash Join (cost=53461.76..89583.48 rows=3738 width=112)
8	Hash Cond: ((vehicles.accident_index)::text = (accidents.accident_index)::text)
9	-> Parallel Seq Scan on vehicle_information vehicles (cost=0.00..35555.05 rows=141144 width=116)
10	Filter: ((age_band_of_driver)::text = '36 - 45'::text)
11	-> Hash (cost=52827.11..52827.11 rows=50772 width=24)
12	-> Seq Scan on accident_information accidents (cost=0.00..52827.11 rows=50772 width=24)
13	Filter: (((urban_or_rural_area)::text = 'Rural'::text) AND (year = 2012))

v) Ποιος είναι ο κατασκευαστής του οχήματος που έχει τα περισσότερα ατυχήματα μεταξύ των ετών 2010 και 2012, η ηλικιακή κατηγορία των οδηγών είναι 26- 35 και η κατηγορία δρόμου είναι Α.





	QUERY PLAN text
1	HashAggregate (cost=107102.34..107272.98 rows=17064 width=29)
2	Group Key: v.make, v.age_band_of_driver, a.first_road_class
3	Filter: (count(v.make) = $0)
4	InitPlan 1 (returns $0)
5	-> Aggregate (cost=4.71..4.72 rows=1 width=8)
6	-> Seq Scan on accident_sum_per_make (cost=0.00..4.17 rows=217 width=8)
7	-> Gather (cost=62351.11..106505.11 rows=47400 width=21)
8	Workers Planned: 3
9	-> Hash Join (cost=61351.11..100765.11 rows=15290 width=21)
10	Hash Cond: ((v.accident_index)::text = (a.accident_index)::text)
11	-> Parallel Seq Scan on vehicle_information v (cost=0.00..35555.05 rows=144281 width=30)
12	Filter: ((age_band_of_driver)::text = '26 - 35'::text)
13	-> Hash (cost=57620.30..57620.30 rows=203185 width=19)
14	-> Seq Scan on accident_information a (cost=0.00..57620.30 rows=203185 width=19)
15	Filter: ((year >= 2010) AND (year <= 2012) AND ((first_road_class)::text = 'A'::text))

Συμπέρασμα

Όπως βλέπουμε υπάρχουν σαφώς πιο εμφανίσιμες βελτιώσεις στο χρόνο εκτέλεσης κάθε ερωτήματος (σε σχέση με το ερώτημα 2) αν και στα δύο τελευταία queries ακόμα δεν παρατηρείται αξιοσημείωτη διαφορά όπως στα άλλα queries. Από την άλλη όμως υπάρχει αξιοσημείωτη διαφορά στο κόστος κάθε query (που το βλέπουμε στο query plan) καθώς τώρα από 2 workers planned ο υπολογιστής μας βάζει 3 (με μέγιστο το 8) προσφέροντας έτσι περισσότερη επεξεργαστική ισχύ και συνεπώς χαμηλότερο κόστος. Άρα συμπεραίνουμε ότι με όλη την επεξεργαστική ισχύ του υπολογιστή μας μπορούν να υπάρξουν βελτιώσεις στο χρόνο εκτέλεσης αλλά κυρίως θα υπάρξουν μεγαλύτερες βελτιώσεις στο κόστος του query.

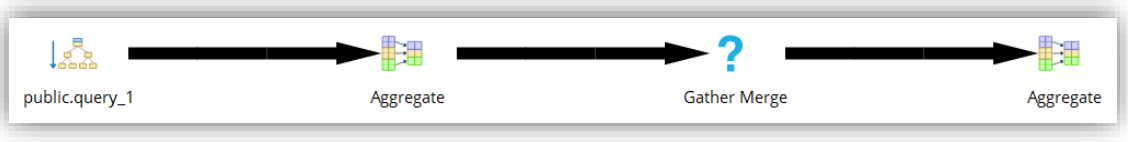
1.4 Ερώτημα Δ

Σε αυτή την ενότητα δημιουργούμε τα κατάλληλα ευρετήρια πάνω στους δύο πίνακες της βάσης δεδομένων, ώστε να επιταχύνουμε την απόκριση του συστήματος. Ας μελετήσουμε ένα προς ένα τα SQL queries του ερωτήματος Α συγκρίνοντας τα αποτελέσματα με βάση το χρόνο.

Ερώτημα 1ο: Βρείτε πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου (road class).

Ανατρέχοντας στο αρχείο d_1_1.sql δημιουργούμε το index query_1 στον πίνακα accident_information και εκτελούμε ξανά το query του αρχείου a_1_1.sql. Το index που χρησιμοποιούμε βασίζεται στη δομή B-Tree για να επιταχύνουμε την ομαδοποίηση του GROUP BY πάνω στο first_road_class.

Γραφικό σχήμα επεξήγησης:



Απλό σχήμα επεξήγησης:

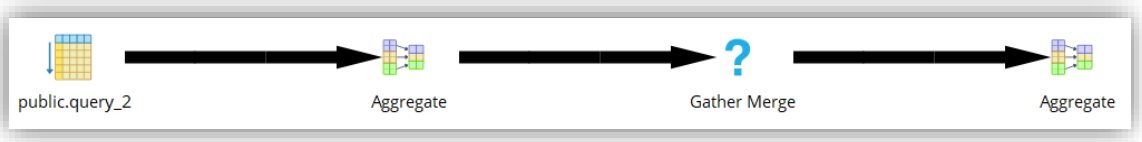
	QUERY PLAN text
1	Finalize GroupAggregate (cost=1000.45..135382.53 rows=6 width=13) (actual time=198.930..483.224 rows=6 loops=1)
2	Group Key: first road class
3	-> Gather Merge (cost=1000.45..135382.41 rows=12 width=13) (actual time=195.847..487.130 rows=18 loops=1)
4	Workers Planned: 2
5	Workers Launched: 2
6	-> Partial GroupAggregate (cost=0.43..134381.00 rows=6 width=13) (actual time=190.158..389.693 rows=6 loops=3)
7	Group Key: first road class
8	-> Parallel Index Only Scan using query 1 on accident information (cost=0.43..130386.62 rows=798864 width=5) (actual time=0.133..299.354 rows=639091 loops=3)
9	Heap Fetches: 300399
10	Planning time: 0.137 ms
11	Execution time: 487.221 ms

Παρατηρούμε ότι από 781ms μειώθηκε ο χρόνος εκτέλεσης σε **487ms**.

Ερώτημα 2ο: Πόσα είναι τα ατυχήματα ανά κατηγορία δρόμου και ανά κατηγορία ατυχήματος (Accident Severity).

Ανατρέχοντας στο αρχείο d_1_2.sql δημιουργούμε το index query_2 στον πίνακα accident_information και εκτελούμε ξανά το query του αρχείου a_1_2.sql. Το index που χρησιμοποιούμε βασίζεται στη δομή B-Tree για να επιταχύνουμε την ομαδοποίηση του GROUP BY πάνω στα first_road_class και accident_severity.

Γραφικό σχήμα εκτέλεσης:



Απλό σχήμα εκτέλεσης:

	QUERY PLAN text
1	Finalize GroupAggregate (cost=1000.45..148680.77 rows=18 width=20) (actual time=112.104..888.915 rows=18 loops=1)
2	Group Key: first road class, accident severity
3	-> Gather Merge (cost=1000.45..148680.32 rows=36 width=20) (actual time=30.504..892.668 rows=52 loops=1)
4	Workers Planned: 2
5	Workers Launched: 2
6	-> Partial GroupAggregate (cost=0.43..147676.14 rows=18 width=20) (actual time=23.605..628.263 rows=17 loops=3)
7	Group Key: first road class, accident severity
8	-> Parallel Index Scan using query 2 on accident information (cost=0.43..141684.48 rows=798864 width=101) (actual time=0.059..529.085 rows=639091 loops=3)
9	Planning time: 0.204 ms
10	Execution time: 892.834 ms

Παρατηρούμε ότι από 256ms αυξήθηκε ο χρόνος εκτέλεσης σε **892ms**.

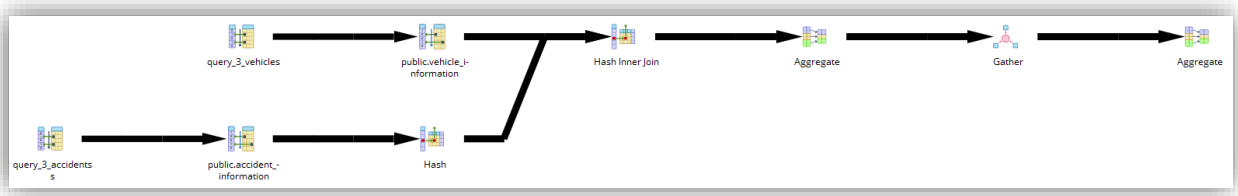
Ερώτημα 3ο: Βρείτε πόσα ατυχήματα έγιναν σε αστική περιοχή πριν από την 1/1/2010 και η ηλικιακή κατηγορία του οδηγού είναι 26-35.

Ανατρέχοντας στο αρχείο d_1_3.sql δημιουργούμε δυο indexes:

- Το query_3_accidents πάνω στον πίνακα accident_information με τα πεδία accident_index, date και urban_or_rural_area. Πρόσθετοι περιορισμοί το urban_or_rural area πρέπει να έχει τη τιμή 'Urban' και το date πρέπει να είναι μικρότερο από '2010-1-1'.
- Το query_3_vehicles πάνω στον πίνακα vehicles_information με τα πεδία accident_index και age_band_of_driver. Πρόσθετος περιορισμός το age_band_of_driver πρέπει να έχει τιμή '26 – 35'.

Εκτελούμε ξανά το query του αρχείου a_1_3.sql. Και στα δυο indexes ακολουθούμε τη δομή B-Tree εφόσον έχουμε συγκρίσεις και ομαδοποίηση με INNER JOIN, WHERE και GROYP BY. Επιπρόσθετα, περιορίζουμε το μέγεθος των index στις τιμές που ενδιαφέρουν το query του ερωτήματος.

Γραφικό σχήμα εκτέλεσης:



Απλό σχήμα εκτέλεσης:

	QUERY PLAN text
1	Finalize GroupAggregate (cost=72939.83..113828.01 rows=36 width=23) (actual time=781.387..781.388 rows=1 loops=1)
2	Group Key: vehicles.age band of driver, accidents.urban or rural area
3	-> Gather (cost=72939.83..113827.11 rows=72 width=23) (actual time=781.373..782.687 rows=3 loops=1)
4	Workers Planned: 2
5	Workers Launched: 2
6	-> Partial GroupAggregate (cost=71939.83..112819.91 rows=36 width=23) (actual time=770.664..770.664 rows=1 loops=3)
7	Group Key: vehicles.age band of driver, accidents.urban or rural area
8	-> Hash Join (cost=71939.83..112387.19 rows=57647 width=108) (actual time=478.547..765.924 rows=28766 loops=3)
9	Hash Cond: ((vehicles.accident index)::text = (accidents.accident index)::text)
10	-> Parallel Bitmap Heap Scan on vehicle information vehicles (cost=11313.95..40422.41 rows=188117 width=116) (actual time=33.489..161.454 rows=150177 loops=3)
11	Recheck Cond: ((age band of driver)::text = '26 - 35'::text)
12	Heap Blocks: exact=16191
13	-> Bitmap Index Scan on query 3 vehicles (cost=0.00..11201.08 rows=451480 width=0) (actual time=31.402..31.402 rows=450531 loops=1)
14	-> Hash (cost=49838.73..49838.73 rows=587532 width=20) (actual time=436.474..436.474 rows=573888 loops=3)
15	Buckets: 65536 Batches: 16 Memory Usage: 2326kB
16	-> Bitmap Heap Scan on accident information accidents (cost=16964.75..49838.73 rows=587532 width=20) (actual time=53.905..248.765 rows=573888 loops=3)
17	Recheck Cond: (((urban or rural area)::text = 'Urban'::text) AND (date < '2010-01-01'::date))
18	Heap Blocks: exact=11226
19	-> Bitmap Index Scan on query 3 accidents (cost=0.00..16817.86 rows=587532 width=0) (actual time=51.452..51.452 rows=573888 loops=3)
20	Planning time: 0.412 ms
21	Execution time: 782.758 ms

Παρατηρούμε ότι από 1s μειώθηκε ο χρόνος εκτέλεσης σε **782ms**.

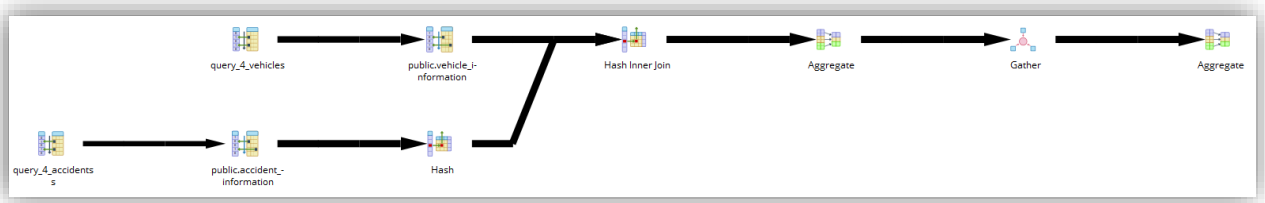
Ερώτημα 4ο: Βρείτε πόσα ατυχήματα έγιναν σε αγροτική περιοχή το έτος 2012 και η ηλικιακή κατηγορία του οδηγού είναι 36-45.

Ανατρέχοντας στο αρχείο d_1_4.sql δημιουργούμε δυο indexes:

- Το query_4_accidents πάνω στον πίνακα accident_information με τα πεδία accident_index, year και urban_or_rural_area. Πρόσθετοι περιορισμοί το urban_or_rural_area πρέπει να έχει τη τιμή 'Rural' και το year πρέπει να είναι '2012'.
- Το query_4_vehicles πάνω στον πίνακα vehicles_information με τα πεδία accident_index και age_band_of_driver. Πρόσθετος περιορισμός το age_band_of_driver πρέπει να έχει τιμή '36 – 45'.

Εκτελούμε ξανά το query του αρχείου a_1_4.sql. Και στα δυο indexes ακολουθούμε τη δομή B-Tree εφόσον έχουμε συγκρίσεις και ομαδοποίηση με INNER JOIN, WHERE και GROYP BY. Επιπρόσθετα, περιορίζουμε το μέγεθος των index στις τιμές που ενδιαφέρουν το query του ερωτήματος.

Γραφικό σχήμα εκτέλεσης:



Απλό σχήμα εκτέλεσης:

	QUERY PLAN text
1	Finalize GroupAggregate (cost=38936.63..68885.20 rows=432 width=27) (actual time=277.748..277.748 rows=1 loops=1)
2	Group Key: vehicles.age band of driver, accidents.urban or rural area, accidents.year
3	-> Gather (cost=38936.63..68872.24 rows=864 width=27) (actual time=277.255..281.267 rows=3 loops=1)
4	Workers Planned: 2
5	Workers Launched: 2
6	-> Partial GroupAggregate (cost=37936.63..67785.84 rows=432 width=27) (actual time=271.616..271.617 rows=1 loops=3)
7	Group Key: vehicles.age band of driver, accidents.urban or rural area, accidents.year
8	-> Hash Join (cost=37936.63..67731.38 rows=5014 width=112) (actual time=186.854..270.507 rows=4184 loops=3)
9	Hash Cond: ((vehicles.accident index)::text = (accidents.accident index)::text)
10	-> Parallel Bitmap Heap Scan on vehicle information vehicles (cost=10941.16..39996.33 rows=183853 width=116) (actual time=30.274..174.287 rows=145229 loops=3)
11	Recheck Cond: ((age band of driver)::text = '36 - 45'::text)
12	Heap Blocks: exact=8569
13	-> Bitmap Index Scan on query 4 vehicles (cost=0.00..10830.85 rows=441247 width=0) (actual time=30.494..30.494 rows=435686 loops=1)
14	-> Hash (cost=26341.88..26341.88 rows=52287 width=24) (actual time=46.587..46.587 rows=50217 loops=3)
15	Buckets: 65536 Batches: 1 Memory Usage: 3357kB
16	-> Bitmap Heap Scan on accident information accidents (cost=1496.57..26341.88 rows=52287 width=24) (actual time=3.388..26.440 rows=50217 loops=3)
17	Recheck Cond: (((urban or rural area)::text = 'Rural'::text) AND (year = 2012))
18	Heap Blocks: exact=1663
19	-> Bitmap Index Scan on query 4 accidents (cost=0.00..1483.50 rows=52287 width=0) (actual time=3.128..3.128 rows=50217 loops=3)
20	Planning time: 0.398 ms
21	Execution time: 281.346 ms

Παρατηρούμε ότι από 616ms μειώθηκε ο χρόνος εκτέλεσης σε **281ms**.

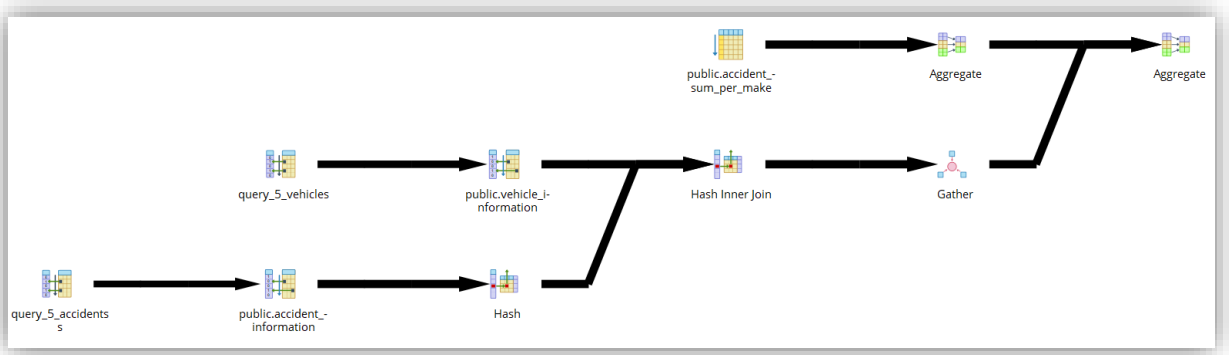
Ερώτημα 5ο: Βρείτε τον κατασκευαστή του οχήματος που έχει τα περισσότερα ατυχήματα μεταξύ των ετών 2010 και 2012, ηλικιακή κατηγορία των οδηγών είναι 26-35 και η κατηγορία δρόμου είναι A.

Ανατρέχοντας στο αρχείο d_1_5.sql δημιουργούμε δυο indexes:

- Το query_5_accidents πάνω στον πίνακα accident_information με τα πεδία accident_index, first_road_class και year. Πρόσθετοι περιορισμοί το first_road_class πρέπει να έχει τη τιμή 'A' και το year να είναι ανάμεσα στο 2010 και 2012.
- Το query_5_vehicles πάνω στον πίνακα vehicles_information με τα πεδία accident_index και age_band_of_driver και make. Πρόσθετος περιορισμός το age_band_of_driver πρέπει να έχει τιμή '26 – 35'.

Εκτελούμε ξανά το query του αρχείου a_1_5.sql. Και στα δυο indexes ακολουθούμε τη δομή B-Tree εφόσον έχουμε συγκρίσεις και ομαδοποίηση με INNER JOIN, WHERE και GROYP BY. Επιπρόσθετα, περιορίζουμε το μέγεθος των index στις τιμές που ενδιαφέρουν το query του ερωτήματος.

Γραφικό σχήμα εκτέλεσης:



Απλό σχήμα εκτέλεσης:

	QUERY PLAN text
1	HashAggregate (cost=10000120689.91..10000120852.63 rows=16272 width=29) (actual time=434.621..434.709 rows=1 loops=1)
2	Group Key: v.make, v.age band of driver, a.first road class
3	Filter: (count(v.make) = $0)
4	Rows Removed by Filter: 216
5	InitPlan 1 (returns $0)
6	-> Aggregate (cost=10000000004.71..10000000004.72 rows=1 width=8) (actual time=0.069..0.069 rows=1 loops=1)
7	-> Seq Scan on accident sum per make (cost=10000000000.00..10000000004.17 rows=217 width=8) (actual time=0.029..0.045 rows=217 loops=1)
8	-> Gather (cost=75135.52..120070.57 rows=49169 width=21) (actual time=255.325..405.439 rows=53584 loops=1)
9	Workers Planned: 2
10	Workers Launched: 2
11	-> Hash Join (cost=74135.52..114153.67 rows=20487 width=21) (actual time=251.306..392.385 rows=17861 loops=3)
12	Hash Cond: ((v.accident index)::text = (a.accident index)::text)
13	-> Parallel Bitmap Heap Scan on vehicle information v (cost=13532.29..48800.51 rows=192018 width=30) (actual time=36.697..138.198 rows=150177 loops=3)
14	Recheck Cond: ((age band of driver)::text = '26 - 35'::text)
15	Heap Blocks: exact=7302
16	-> Bitmap Index Scan on query 5 vehicles (cost=0.00..13417.08 rows=460842 width=0) (actual time=35.489..35.489 rows=450531 loops=1)
17	-> Hash (cost=56847.21..56847.21 rows=204562 width=19) (actual time=155.789..155.789 rows=207329 loops=3)
18	Buckets: 65536 Batches: 4 Memory Usage: 2948kB
19	-> Bitmap Heap Scan on accident information a (cost=5146.37..56847.21 rows=204562 width=19) (actual time=16.071..86.808 rows=207329 loops=3)
20	Recheck Cond: (((first road class)::text = 'A'::text) AND (year >= 2010) AND (year <= 2012))
21	Heap Blocks: exact=5645
22	-> Bitmap Index Scan on query 5 accidents (cost=0.00..5095.23 rows=204562 width=0) (actual time=15.126..15.126 rows=207329 loops=3)
23	Planning time: 0.514 ms
24	Execution time: 435.351 ms

Παρατηρούμε ότι από 781ms μειώθηκε ο χρόνος εκτέλεσης σε **435ms**.

1.5 Παρατηρήσεις

Όλα τα index που δημιουργήθηκαν σε αυτό το κομμάτι της εργασίας μπορούν εύκολα θα διαγραφούν, εκτελώντας τις εντολές του αρχείου **D_indexes/reset.sql**.

1.6 Ερώτημα Ε

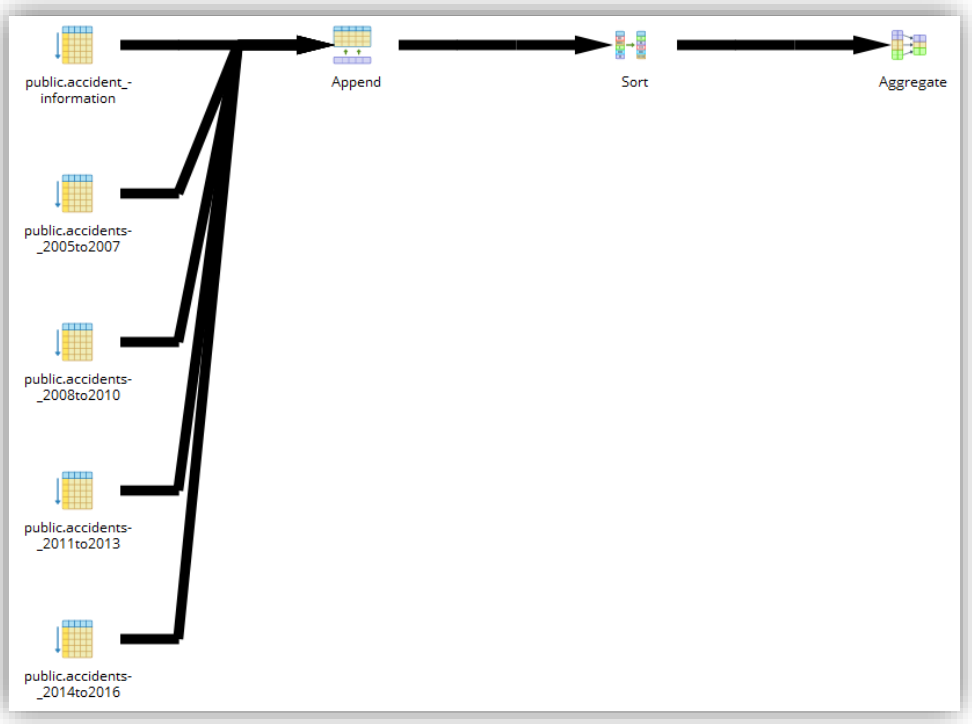
Έχοντας υπ' όψη τα προηγούμενα ερωτήματα, χωρίζουμε τους δυο πίνακες accident_information και vehicle_information σε υποπίνακες με βάση την κληρονομικότητα πινάκων. Επιλέξαμε να κάνουμε το partitioning με βάση ενός range και για τις δυο περιπτώσεις. Στην περίπτωση του accident_information δημιουργούμε υποπίνακες που χωρίζονται χρονικά πάνω στο πεδίο year και για το vehicle_information χωρίζουμε σε υποπίνακες πάνω στο sex_of_driver.

Ο λόγος για τον οποίο δεν επιλέγουμε άλλους τρόπους partitioning όπως hashing, random ή list είναι γιατί τα δεδομένα έχουν ήδη τα πεδία year και sex_of_driver που παραπέμπουν σε μια άτυπη κατηγοριοποίηση. Επομένως, είναι εύλογο και απλό να γίνει χωρισμός πάνω σε χρονικές περιόδους και πάνω στο φύλο των οδηγών.

Αξίζει να σημειωθεί ότι η postgresql μας δίνει και έναν εναλλακτικό τρόπο partitioning με τη χρήση PARTITION BY RANGE. Για τις ανάγκες της εργασίας, έχουμε χρησιμοποιήσει κληρονομικότητα για κάθε υποπίνακα καθώς και triggers με functions ώστε κατά την εισαγωγή των δεδομένων να γίνει ανακατεύθυνση και στους αντίστοιχους πίνακες του partitioning. Τέλος, δημιουργήσαμε δυο index για το year όσο αφορά τα partitions του accident_information και για το sex_of_driver όσον αφορά τα partitions του vehicle_information.

Ερώτημα 1ο: Βρείτε πόσα ατυχήματα έγιναν ανά κατηγορία δρόμου (road class).

Γραφικό σχήμα εκτέλεσης:

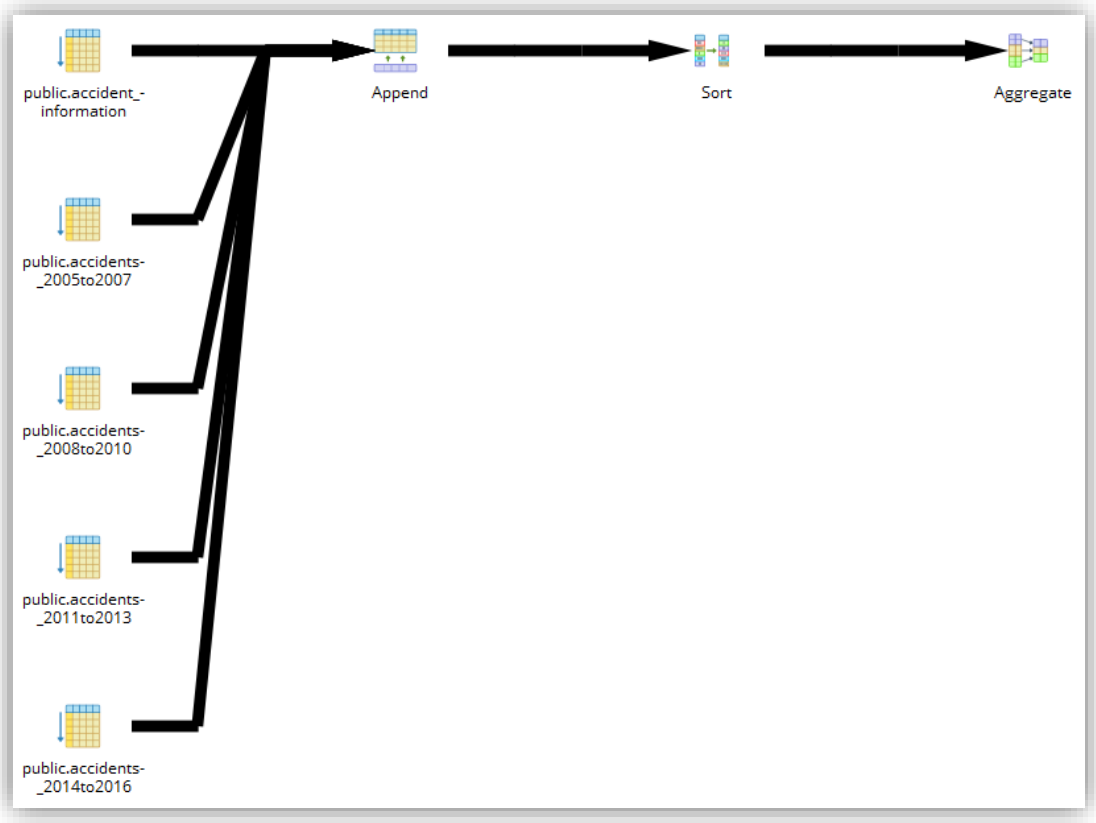


Απλό σχήμα εκτέλεσης:

	QUERY PLAN text
1	GroupAggregate (cost=50000269533.39..50000283914.95 rows=200 width=13) (actual time=1285.341..1594.729 rows=6 loops=1)
2	Group Key: accident information.first road class
3	-> Sort (cost=50000269533.39..50000274326.57 rows=1917275 width=5) (actual time=1023.442..1319.495 rows=1917274 loops=1)
4	Sort Key: accident information.first road class
5	Sort Method: external merge Disk: 29088kB
6	-> Append (cost=10000000000.00..50000043244.74 rows=1917275 width=5) (actual time=0.014..385.210 rows=1917274 loops=1)
7	-> Seq Scan on accident information (cost=10000000000.00..10000000000.00 rows=1 width=5) (actual time=0.003..0.003 rows=0 loops=1)
8	-> Seq Scan on accidents 2005to2007 (cost=10000000000.00..10000012862.11 rows=570011 width=5) (actual time=0.010..88.045 rows=570011 loops=1)
9	-> Seq Scan on accidents 2008to2010 (cost=10000000000.00..10000011012.59 rows=488559 width=5) (actual time=0.013..67.900 rows=488559 loops=1)
10	-> Seq Scan on accidents 2011to2013 (cost=10000000000.00..10000009823.05 rows=435705 width=5) (actual time=0.010..63.227 rows=435705 loops=1)
11	-> Seq Scan on accidents 2014to2016 (cost=10000000000.00..10000009546.99 rows=422999 width=5) (actual time=0.009..59.527 rows=422999 loops=1)
12	Planning time: 0.256 ms
13	Execution time: 1599.623 ms

Ερώτημα 2ο: Πόσα είναι τα ατυχήματα ανά κατηγορία δρόμου και ανά κατηγορία ατυχήματος (Accident Severity).

Γραφικό σχήμα εκτέλεσης:

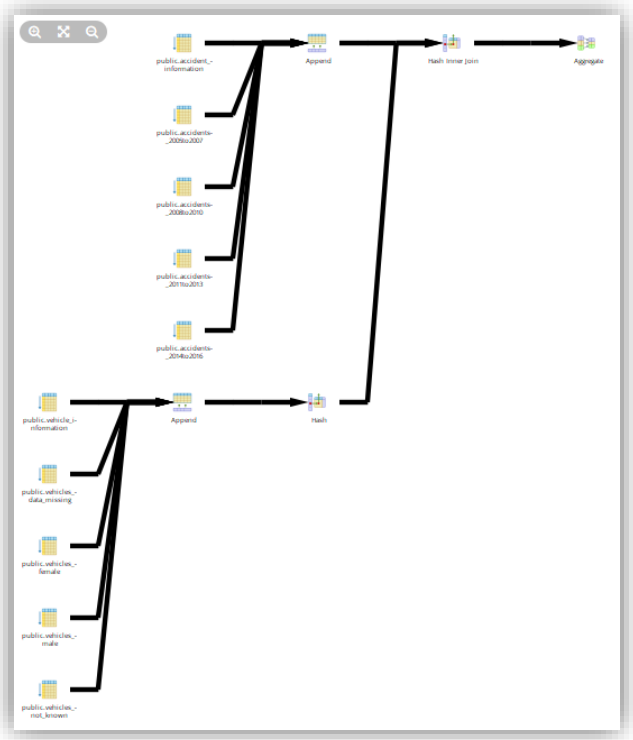


Απλό σχήμα εκτέλεσης:

	QUERY PLAN text
1	GroupAggregate (cost=50000361282.39..50000380855.14 rows=40000 width=20) (actual time=2421.815..3283.000 rows=18 loops=1)
2	Group Key: accident information.first road class, accident information.accident severity
3	-> Sort (cost=50000361282.39..50000366075.57 rows=1917275 width=44) (actual time=2415.325..2913.515 rows=1917274 loops=1)
4	Sort Key: accident information.first road class, accident information.accident severity
5	Sort Method: external merge Disk: 213552kB
6	-> Append (cost=10000000000.00..50000043244.74 rows=1917275 width=44) (actual time=0.069..867.953 rows=1917274 loops=1)
7	-> Seq Scan on accident information (cost=10000000000.00..10000000000.00 rows=1 width=101) (actual time=0.002..0.002 rows=0 loops=1)
8	-> Seq Scan on accidents 2005to2007 (cost=10000000000.00..10000012862.11 rows=570011 width=44) (actual time=0.067..226.469 rows=570011 loops=1)
9	-> Seq Scan on accidents 2008to2010 (cost=10000000000.00..10000011012.59 rows=488559 width=44) (actual time=0.040..190.949 rows=488559 loops=1)
10	-> Seq Scan on accidents 2011to2013 (cost=10000000000.00..10000009823.05 rows=435705 width=44) (actual time=0.042..170.226 rows=435705 loops=1)
11	-> Seq Scan on accidents 2014to2016 (cost=10000000000.00..10000009546.99 rows=422999 width=44) (actual time=0.042..171.005 rows=422999 loops=1)
12	Planning time: 0.184 ms
13	Execution time: 3313.916 ms

Ερώτημα 3ο: Βρείτε πόσα ατυχήματα έγιναν σε αστική περιοχή πριν από την 1/1/2010 και η ηλικιακή κατηγορία του οδηγού είναι 26-35.

Γραφικό σχήμα εκτέλεσης:

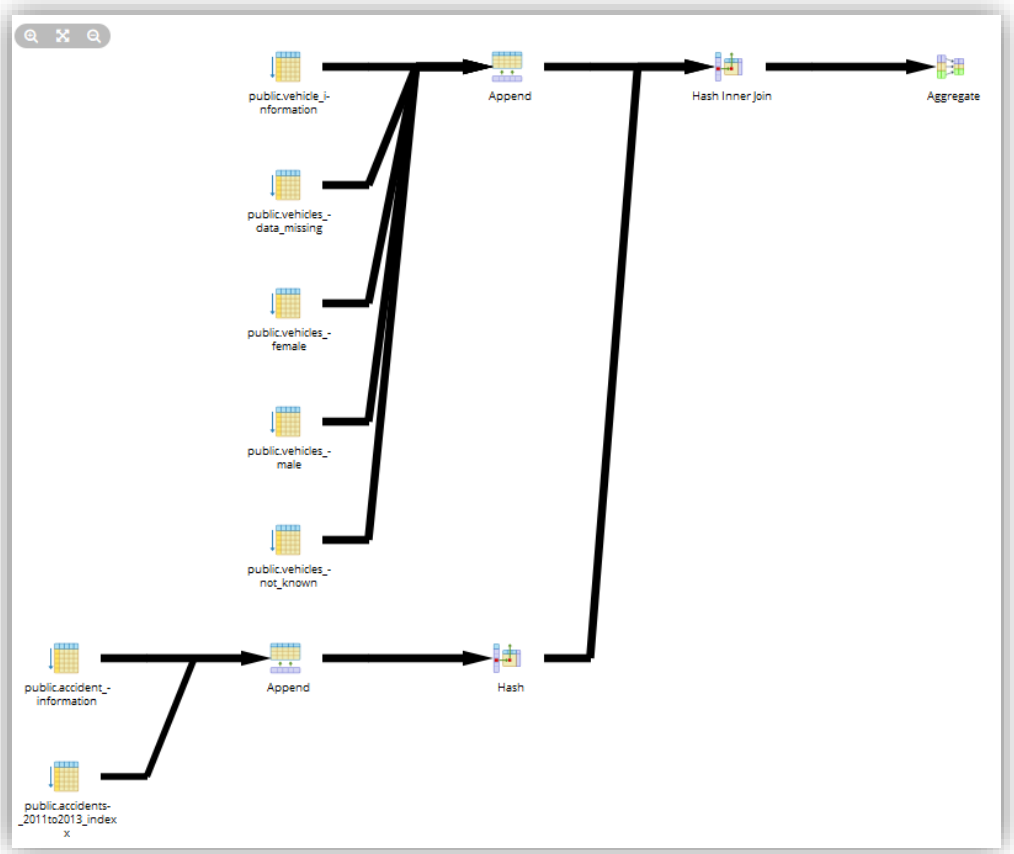


Απλό σχήμα εκτέλεσης:

	QUERY PLAN text
1	GroupAggregate (cost=60000063961.07..100054995615.60 rows=40000 width=23) (actual time=1237.679..1237.679 rows=1 loops=1)
2	Group Key: vehicles.age band of driver, accidents.urban or rural area
3	-> Hash Join (cost=60000063961.07..100045312992.77 rows=1290963044 width=47) (actual time=621.074..1225.385 rows=86298 loops=1)
4	Hash Cond: ((accidents.accident index)::text = (vehicles.accident index)::text)
5	-> Append (cost=10000000000.00..50000052831.11 rows=575602 width=20) (actual time=0.020..303.501 rows=573888 loops=1)
6	-> Seq Scan on accident information accidents (cost=10000000000.00..10000000000.00 rows=1 width=20) (actual time=0.008..0.008 rows=0 loops=1)
7	Filter: ((date < '2010-01-01'::date) AND ((urban or rural area)::text = 'Urban'::text))
8	-> Seq Scan on accidents 2005to2007 accidents 1 (cost=10000000000.00..10000015712.17 rows=361736 width=20) (actual time=0.011..102.967 rows=360761 loops=1)
9	Filter: ((date < '2010-01-01'::date) AND ((urban or rural area)::text = 'Urban'::text))
10	Rows Removed by Filter: 209250
11	-> Seq Scan on accidents 2008to2010 accidents 2 (cost=10000000000.00..10000013455.39 rows=213816 width=20) (actual time=0.042..77.511 rows=213127 loops=1)
12	Filter: ((date < '2010-01-01'::date) AND ((urban or rural area)::text = 'Urban'::text))
13	Rows Removed by Filter: 275432
14	-> Seq Scan on accidents 2011to2013 accidents 3 (cost=10000000000.00..10000012001.58 rows=25 width=20) (actual time=45.721..45.721 rows=0 loops=1)
15	Filter: ((date < '2010-01-01'::date) AND ((urban or rural area)::text = 'Urban'::text))
16	Rows Removed by Filter: 435705
17	-> Seq Scan on accidents 2014to2016 accidents 4 (cost=10000000000.00..10000011661.99 rows=24 width=20) (actual time=45.353..45.353 rows=0 loops=1)
18	Filter: ((date < '2010-01-01'::date) AND ((urban or rural area)::text = 'Urban'::text))
19	Rows Removed by Filter: 422999
20	-> Hash (cost=50000053973.06..50000053973.06 rows=448561 width=55) (actual time=620.833..620.833 rows=450531 loops=1)
21	Buckets: 65536 (originally 65536) Batches: 32 (originally 16) Memory Usage: 3585kB
22	-> Append (cost=10000000000.00..50000053973.06 rows=448561 width=55) (actual time=0.043..469.984 rows=450531 loops=1)
23	-> Seq Scan on vehicle information vehicles (cost=10000000000.00..10000000000.00 rows=1 width=116) (actual time=0.002..0.002 rows=0 loops=1)
24	Filter: ((age band of driver)::text = '26 - 35'::text)
25	-> Seq Scan on vehicles data missing vehicles 1 (cost=10000000000.00..10000000002.85 rows=2 width=72) (actual time=0.041..0.054 rows=2 loops=1)
26	Filter: ((age band of driver)::text = '26 - 35'::text)
27	Rows Removed by Filter: 66
28	-> Seq Scan on vehicles female vehicles 2 (cost=10000000000.00..10000015426.56 rows=140105 width=54) (actual time=0.026..140.639 rows=141470 loops=1)
29	Filter: ((age band of driver)::text = '26 - 35'::text)
30	Rows Removed by Filter: 491535
31	-> Seq Scan on vehicles male vehicles 3 (cost=10000000000.00..10000036518.01 rows=305654 width=55) (actual time=0.043..289.084 rows=306282 loops=1)
32	Filter: ((age band of driver)::text = '26 - 35'::text)
33	Rows Removed by Filter: 1161799
34	-> Seq Scan on vehicles not known vehicles 4 (cost=10000000000.00..10000002025.64 rows=2799 width=73) (actual time=0.055..11.854 rows=2777 loops=1)
35	Filter: ((age band of driver)::text = '26 - 35'::text)
36	Rows Removed by Filter: 73274
37	Planning time: 0.837 ms
38	Execution time: 1237.778 ms

Ερώτημα 4ο: Βρείτε πόσα ατυχήματα έγιναν σε αγροτική περιοχή το έτος 2012 και η ηλικιακή κατηγορία του οδηγού είναι 36-45.

Γραφικό σχήμα εκτέλεσης:

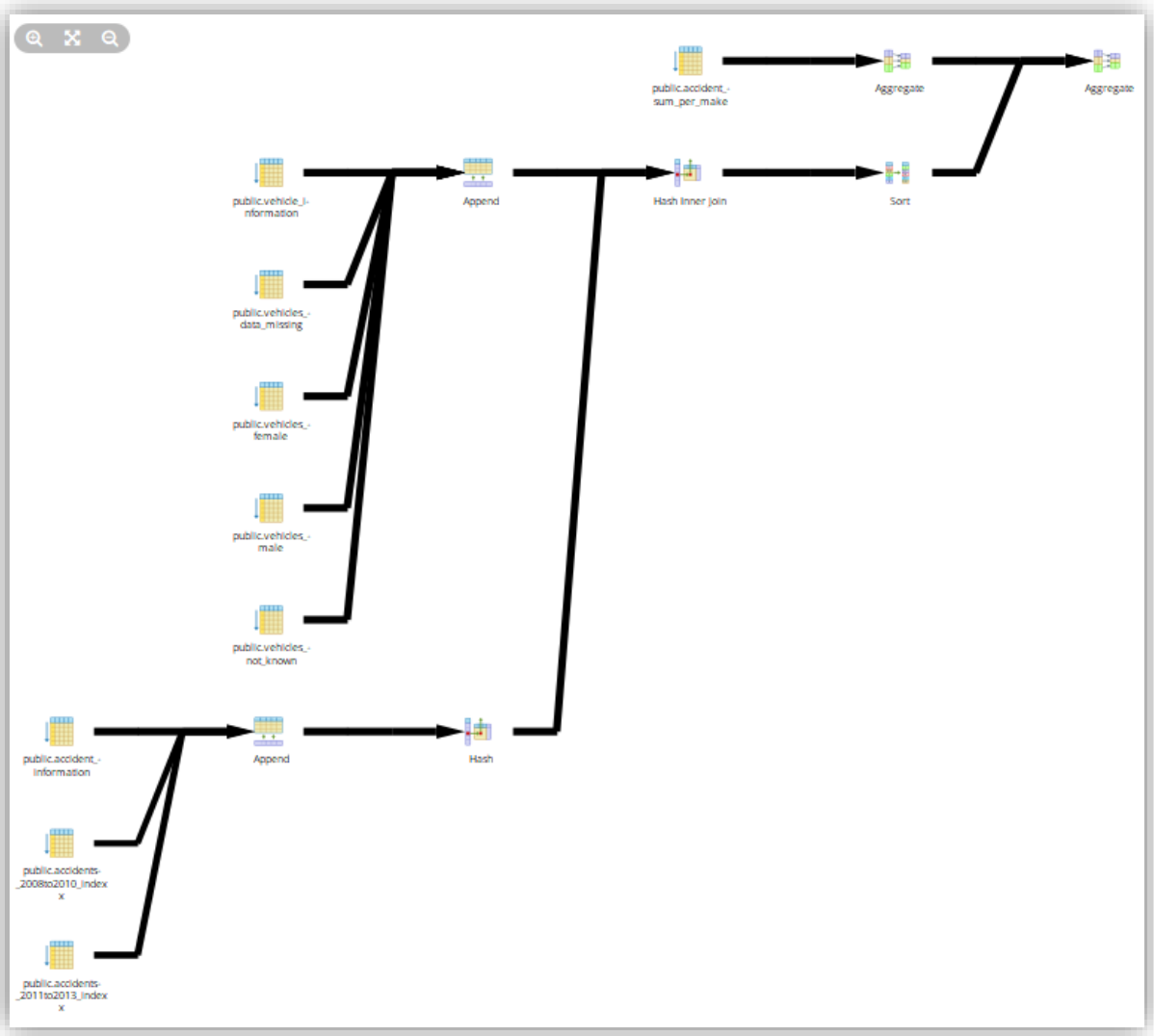


Απλό σχήμα εκτέλεσης:

	QUERY PLAN text
1	Finalize GroupAggregate (cost=77392.53..1466548.63 rows=1005400 width=27) (actual time=1538.885..1538.887 rows=1 loops=1)
2	Group Key: vehicles.age band of driver, accidents.urban or rural area, accidents.year
3	-> Gather (cost=77392.53..1436386.63 rows=2010800 width=27) (actual time=1499.896..1543.611 rows=3 loops=1)
4	Workers Planned: 2
5	Workers Launched: 2
6	-> Partial GroupAggregate (cost=76392.53..1234306.63 rows=1005400 width=27) (actual time=1441.916..1441.916 rows=1 loops=3)
7	Group Key: vehicles.age band of driver, accidents.urban or rural area, accidents.year
8	-> Merge Join (cost=76392.53..765850.36 rows=45840227 width=51) (actual time=1408.702..1441.166 rows=4184 loops=3)
9	Merge Cond: ((vehicles.accident index)::text = (accidents.accident index)::text)
10	-> Sort (cost=60465.69..60922.27 rows=182633 width=55) (actual time=854.974..907.043 rows=93799 loops=3)
11	Sort Key: vehicles.accident index
12	Sort Method: external merge Disk: 13184kB
13	-> Append (cost=0.00..38260.85 rows=182633 width=55) (actual time=0.115..220.745 rows=145229 loops=3)
14	-> Parallel Seq Scan on vehicle information vehicles (cost=0.00..0.00 rows=1 width=116) (actual time=0.001..0.001 rows=0 loops=3)
15	Filter: ((age band of driver)::text = '36 - 45'::text)
16	-> Parallel Seq Scan on vehicles data missing vehicles 1 (cost=0.00..2.50 rows=1 width=72) (actual time=0.009..0.009 rows=0 loops=3)
17	Filter: ((age band of driver)::text = '36 - 45'::text)
18	Rows Removed by Filter: 23
19	-> Parallel Seq Scan on vehicles female vehicles 2 (cost=0.00..10810.90 rows=56443 width=54) (actual time=0.104..70.286 rows=45117 loops=3)
20	Filter: ((age band of driver)::text = '36 - 45'::text)
21	Rows Removed by Filter: 165884
22	-> Parallel Seq Scan on vehicles male vehicles 3 (cost=0.00..25813.26 rows=125338 width=55) (actual time=0.060..134.719 rows=99654 loops=3)
23	Filter: ((age band of driver)::text = '36 - 45'::text)
24	Rows Removed by Filter: 389707
25	-> Parallel Seq Scan on vehicles not known vehicles 4 (cost=0.00..1634.20 rows=850 width=73) (actual time=0.029..3.700 rows=458 loops=3)
26	Filter: ((age band of driver)::text = '36 - 45'::text)
27	Rows Removed by Filter: 24893
28	-> Sort (cost=15926.84..16052.51 rows=50268 width=24) (actual time=491.529..497.685 rows=50495 loops=3)
29	Sort Key: accidents.accident index
30	Sort Method: external sort Disk: 1968kB
31	-> Append (cost=0.00..12001.58 rows=50268 width=24) (actual time=18.712..111.491 rows=50217 loops=3)
32	-> Seq Scan on accident information accidents (cost=0.00..0.00 rows=1 width=24) (actual time=0.017..0.017 rows=0 loops=3)
33	Filter: (((urban or rural area)::text = 'Rural'::text) AND (year = 2012))
34	-> Seq Scan on accidents 2011to2013 accidents 1 (cost=0.00..12001.58 rows=50267 width=24) (actual time=18.693..106.991 rows=50217 loops=3)
35	Filter: (((urban or rural area)::text = 'Rural'::text) AND (year = 2012))
36	Rows Removed by Filter: 385488
37	Planning time: 0.619 ms
38	Execution time: 1547.125 ms

Ερώτημα 5ο: Βρείτε τον κατασκευαστή του οχήματος που έχει τα περισσότερα ατυχήματα μεταξύ των ετών 2010 και 2012, ηλικιακή κατηγορία των οδηγών είναι 26-35 και η κατηγορία δρόμου είναι Α.

Γραφικό σχήμα εκτέλεσης:



Απλό σχήμα εκτέλεσης:

	QUERY PLAN text
1	Sort (cost=47655999.07..47675999.07 rows=8000000 width=29) (actual time=3009.881..3009.890 rows=217 loops=1)
2	Sort Key: (count(v.make)) DESC, v.make
3	Sort Method: quicksort Memory: 41kB
4	-> Finalize GroupAggregate (cost=41759125.88..46355920.33 rows=8000000 width=29) (actual time=3008.247..3009.758 rows=217 loops=1)
5	Group Key: v.make, v.age band of driver, a.first road class
6	-> Gather Merge (cost=41759125.88..46115920.33 rows=16000000 width=29) (actual time=3008.241..3009.706 rows=391 loops=1)
7	Workers Planned: 2
8	Workers Launched: 2
9	-> Partial GroupAggregate (cost=41758125.85..44268123.30 rows=8000000 width=29) (actual time=2781.821..2786.801 rows=130 loops=3)
10	Group Key: v.make, v.age band of driver, a.first road class
11	-> Sort (cost=41758125.85..42244125.34 rows=194399796 width=21) (actual time=2781.814..2783.446 rows=17861 loops=3)
12	Sort Key: v.make
13	Sort Method: quicksort Memory: 570kB
14	-> Merge Join (cost=109564.20..3034516.12 rows=194399796 width=21) (actual time=2620.785..2774.236 rows=17861 loops=3)
15	Merge Cond: ((v.accident index)::text = (a.accident index)::text)
16	-> Sort (cost=59154.82..59623.27 rows=187381 width=30) (actual time=890.535..935.980 rows=90164 loops=3)
17	Sort Key: v.accident index
18	Sort Method: external merge Disk: 4608kB
19	-> Append (cost=0.00..38260.85 rows=187381 width=30) (actual time=0.024..180.138 rows=150177 loops=3)
20	-> Parallel Seq Scan on vehicle information v (cost=0.00..0.00 rows=1 width=30) (actual time=0.000..0.000 rows=0 loops=3)
21	Filter: ((age band of driver)::text = '26 - 35'::text)
22	-> Parallel Seq Scan on vehicles data missing v 1 (cost=0.00..2.50 rows=1 width=47) (actual time=0.003..0.006 rows=1 loops=3)
23	Filter: ((age band of driver)::text = '26 - 35'::text)
24	Rows Removed by Filter: 22
25	-> Parallel Seq Scan on vehicles female v 2 (cost=0.00..10810.90 rows=58377 width=29) (actual time=0.021..59.964 rows=47157 loops=3)
26	Filter: ((age band of driver)::text = '26 - 35'::text)
27	Rows Removed by Filter: 163845
28	-> Parallel Seq Scan on vehicles male v 3 (cost=0.00..25813.26 rows=127356 width=30) (actual time=0.025..101.455 rows=102094 loops=3)
29	Filter: ((age band of driver)::text = '26 - 35'::text)
30	Rows Removed by Filter: 387266
31	-> Parallel Seq Scan on vehicles not known v 4 (cost=0.00..1634.20 rows=1646 width=48) (actual time=0.002..5.676 rows=926 loops=3)
32	Filter: ((age band of driver)::text = '26 - 35'::text)
33	Rows Removed by Filter: 24425
34	-> Materialize (cost=50409.38..51449.51 rows=208025 width=19) (actual time=1693.267..1742.386 rows=208463 loops=3)
35	-> Sort (cost=50409.38..50929.44 rows=208025 width=19) (actual time=1693.264..1724.588 rows=207329 loops=3)
36	Sort Key: a.accident index
37	Sort Method: external merge Disk: 5280kB
38	-> Append (cost=0.00..27767.62 rows=208025 width=19) (actual time=20.535..234.392 rows=207329 loops=3)
39	-> Seq Scan on accident information a (cost=0.00..0.00 rows=1 width=19) (actual time=0.016..0.016 rows=0 loops=3)
40	Filter: ((year >= 2010) AND (year <= 2012) AND ((first road class)::text = 'A'::text))
41	-> Seq Scan on accidents 2008to2010 a 1 (cost=0.00..14676.78 rows=70483 width=19) (actual time=20.518..112.641 rows=70274 loops=3)
42	Filter: ((year >= 2010) AND (year <= 2012) AND ((first road class)::text = 'A'::text))
43	Rows Removed by Filter: 418285
44	-> Seq Scan on accidents 2011to2013 a 2 (cost=0.00..13090.84 rows=137541 width=19) (actual time=0.025..104.305 rows=137055 loops=3)
45	Filter: ((year >= 2010) AND (year <= 2012) AND ((first road class)::text = 'A'::text))
46	Rows Removed by Filter: 298650
47	Planning time: 0.370 ms
48	Execution time: 3011.815 ms

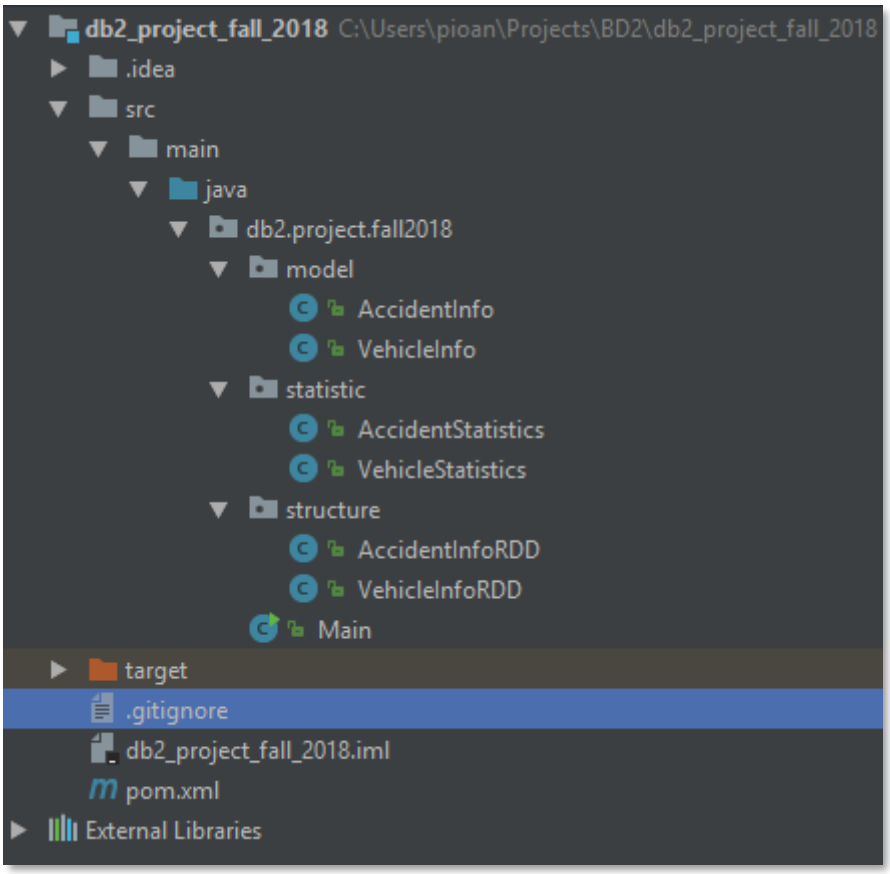
1.7 Παρατηρήσεις

Όλα τα tables, indexes, functions και triggers που δημιουργήθηκαν σε αυτό το κομμάτι της εργασίας μπορούν εύκολα θα διαγραφούν, εκτελώντας τις εντολές του αρχείου **E_partitions/reset.sql**. Υπάρχει περίπτωση τα δεδομένα της εργασίας να πρέπει να ξαναεισαχθούν μετά από αυτές τις ενέργειες.

2 Μέρος 2

2.1 Εισαγωγή

Η δομή του project είναι αυτή που παρουσιάζεται στην εικόνα 2.1 και επίσης είναι αντικειμενοστραφές.



Εικόνα 2.1

Η εργασία είναι ένα maven project και τα dependencies βρίσκονται στο αρχείο pom.xml.

Ο φάκελος **model** περιέχει 2 αρχεία. Το αρχείο AccidentInfo θα αντιπροσωπεύει κάθε εγγραφή από τον πίνακα accidents_info, ενώ το αρχείο VehicleInfo θα αντιπροσωπεύει κάθε εγγραφή από τον πίνακα vehicle_info.

Στο φάκελο **statistic** περιέχονται τα αρχεία AccidentStatistics και VehicleStatistics. Στα αρχεία αυτά βρίσκονται οι κολάσεις που περιέχουν τις μεθόδους για την παραγωγή των στατιστικών.

Τέλος, στο φάκελο **structure** υπάρχουν τα αρχεία AccidentInfoRDD και VehicleInfoRDD τα οποία είναι υπεύθυνα για την φόρτωση των δεδομένων και την δημιουργία των δομών που χρειαζόμαστε.

Για να φορτωθούν τα δεδομένα από τα αρχεία .csv θα πρέπει να ορίσουμε το absolute path στο οποίο βρίσκονται. Όπως φαίνεται και στην εικόνα 2.2 και 2.3 . Επομένως για να φορτωθούν τα δεδομένα θα πρέπει να εισάγετε, στα αρχεία **AccidentInfoRDD** και **VehicleInfoRDD** ,το αντίστοιχο absolute path τον αρχείων που βρίσκονται στον υπολογιστή σας.

```
private AccidentInfoRDD() {
    accidentCSV = config.textFile( path: "/home/panos/Projects/BD2/db2_project_fall_2018/src/main/java/db2/project/fall2018/csv/db2_Accident_Information.csv", minPartition
}
```

Εικόνα 2.2

```
private VehicleInfoRDD( ) {
    vehicleCSV = config.textFile( path: "/home/panos/Projects/BD2/db2_project_fall_2018/src/main/java/db2/project/fall2018/csv/db2_Vehicle_Information.csv", minPartitions:
}
```

Εικόνα 2.3

Το αρχείο εκκίνησης είναι το **Main**. Οι μέθοδοι που παράγουν τα στατιστικά είναι commented. Έτσι οποιαδήποτε μέθοδο θέλετε να εκτελέσετε τη κάνετε uncomment και εκκινείτε την εφαρμογή. Τα αποτελέσματα θα εμφανίζονται στην κονσόλα, όπως γίνεται και στα επόμενα παραδείγματα.

2.2 Ερώτημα Α

Ακολουθώντας τις οδηγίες ,παρουσιάζονται κάποια στατιστικά δεδομένα.

1. Στην εικόνα 2.4 παρουσιάζεται ο συνολικός αριθμός των ατυχημάτων που συνέβησαν. Όπως φαίνεται ο συνολικός αριθμός των ατυχημάτων είναι 1917274.

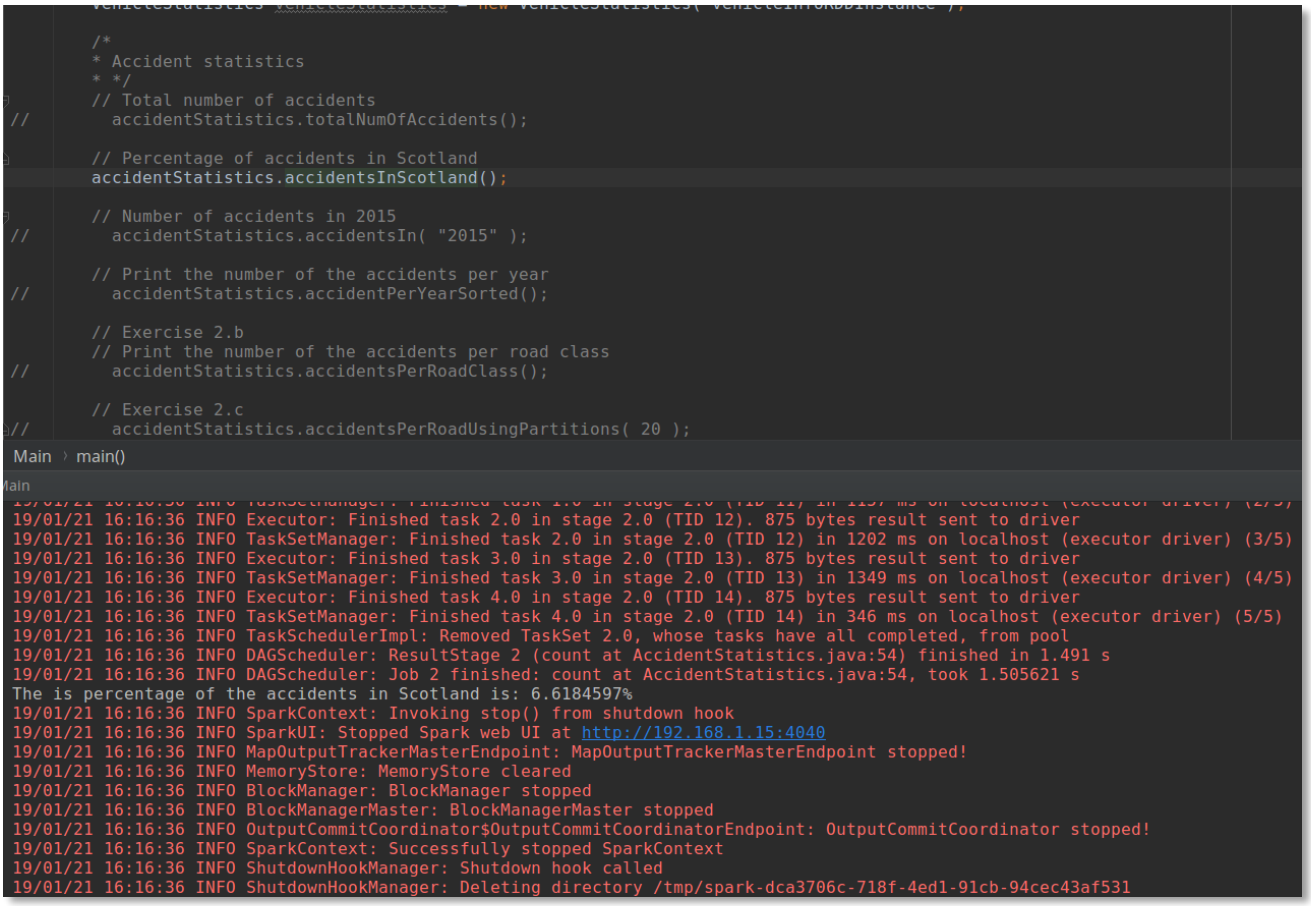
```
        * */
        VehicleInfoRDD.setConfig( jsc );
        // Generate an instance of vehicle info data
        VehicleInfoRDD vehicleInfoRDDInstance   = VehicleInfoRDD.getInstance();
        // Vehicle statistics obj
        VehicleStatistics vehicleStatistics = new VehicleStatistics( vehicleInfoRDDInstance );

        /*
        * Accident statistics
        * */
        // Total number of accidents
        accidentStatistics.totalNumOfAccidents();

        // Percentage of accidents in Scotland
//        accidentStatistics.accidentsInScotland();

        // Number of accidents in 2015
//        accidentStatistics.accidentsIn( "2015" );

        // Print the number of the accidents per year
//        accidentStatistics.accidentPerYearSorted();
```

Main › main()

```
Main
19/01/21 16:12:49 INFO Executor: Finished task 0.0 in stage 1.0 (TID 5). 875 bytes result sent to driver
19/01/21 16:12:49 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 5) in 1216 ms on localhost (executor driver) (3/5)
19/01/21 16:12:49 INFO Executor: Finished task 1.0 in stage 1.0 (TID 6). 875 bytes result sent to driver
19/01/21 16:12:49 INFO TaskSetManager: Finished task 1.0 in stage 1.0 (TID 6) in 1248 ms on localhost (executor driver) (4/5)
19/01/21 16:12:49 INFO Executor: Finished task 4.0 in stage 1.0 (TID 9). 875 bytes result sent to driver
19/01/21 16:12:49 INFO TaskSetManager: Finished task 4.0 in stage 1.0 (TID 9) in 444 ms on localhost (executor driver) (5/5)
19/01/21 16:12:49 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
19/01/21 16:12:49 INFO DAGScheduler: ResultStage 1 (count at VehicleStatistics.java:18) finished in 1.440 s
19/01/21 16:12:49 INFO DAGScheduler: Job 1 finished: count at VehicleStatistics.java:18, took 1.462801 s
The total number of accidents is: 1917274
19/01/21 16:12:49 INFO SparkContext: Invoking stop() from shutdown hook
19/01/21 16:12:49 INFO SparkUI: Stopped Spark web UI at http://192.168.1.15:4040
19/01/21 16:12:49 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
19/01/21 16:12:49 INFO MemoryStore: MemoryStore cleared
19/01/21 16:12:49 INFO BlockManager: BlockManager stopped
19/01/21 16:12:49 INFO BlockManagerMaster: BlockManagerMaster stopped
19/01/21 16:12:49 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
19/01/21 16:12:49 INFO SparkContext: Successfully stopped SparkContext
19/01/21 16:12:49 INFO ShutdownHookManager: Shutdown hook called
19/01/21 16:12:49 INFO ShutdownHookManager: Deleting directory /tmp/spark-1b859211-30f4-46ef-91e2-ce7d95d44861
```

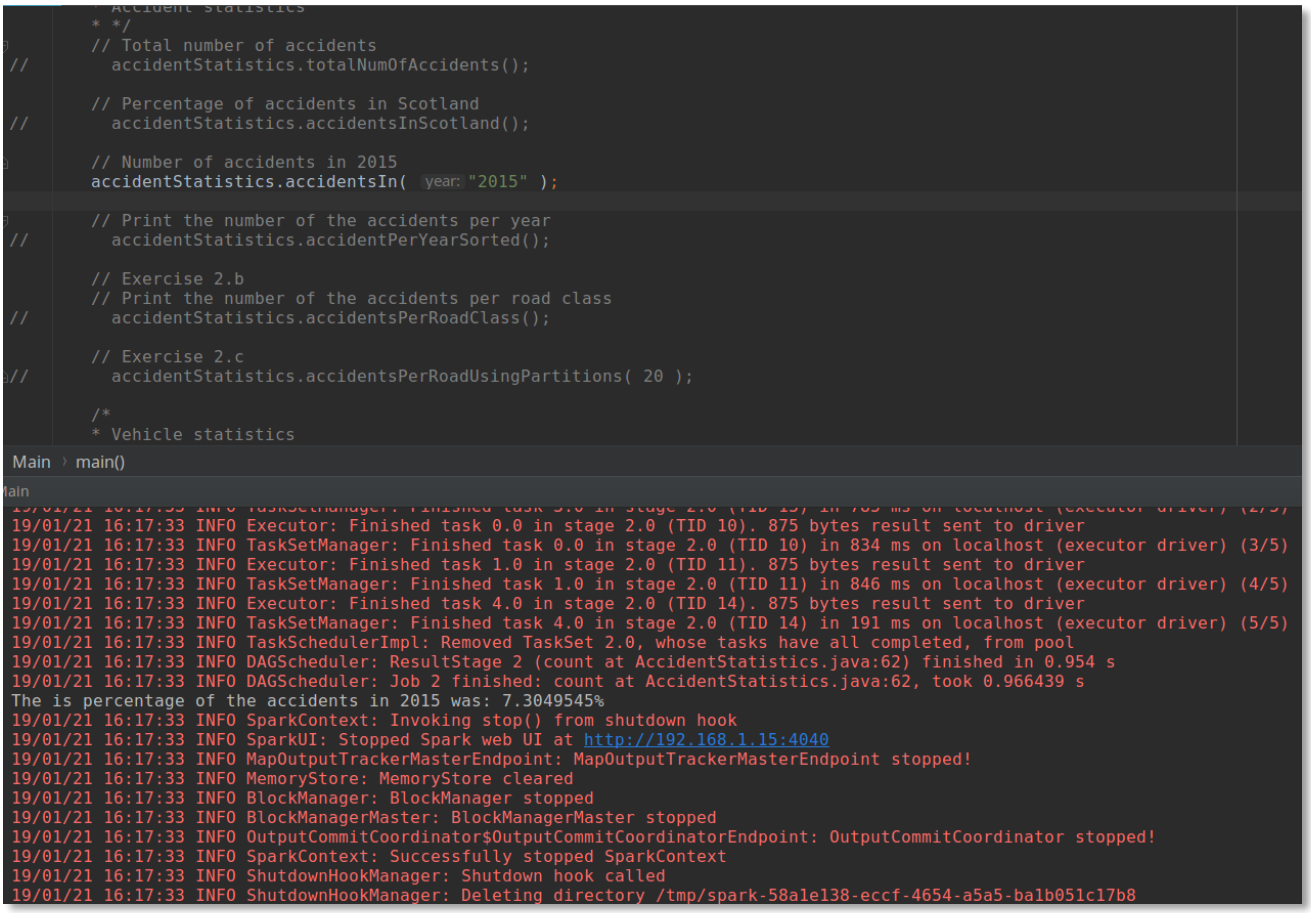
Εικόνα 2.4

2. Στην εικόνα 2.5 παρουσιάζεται το ποσοστό επι τις εκατό των ατυχημάτων που συνέβησαν στη Σκωτία, το οποίο είναι 6,6184597%.



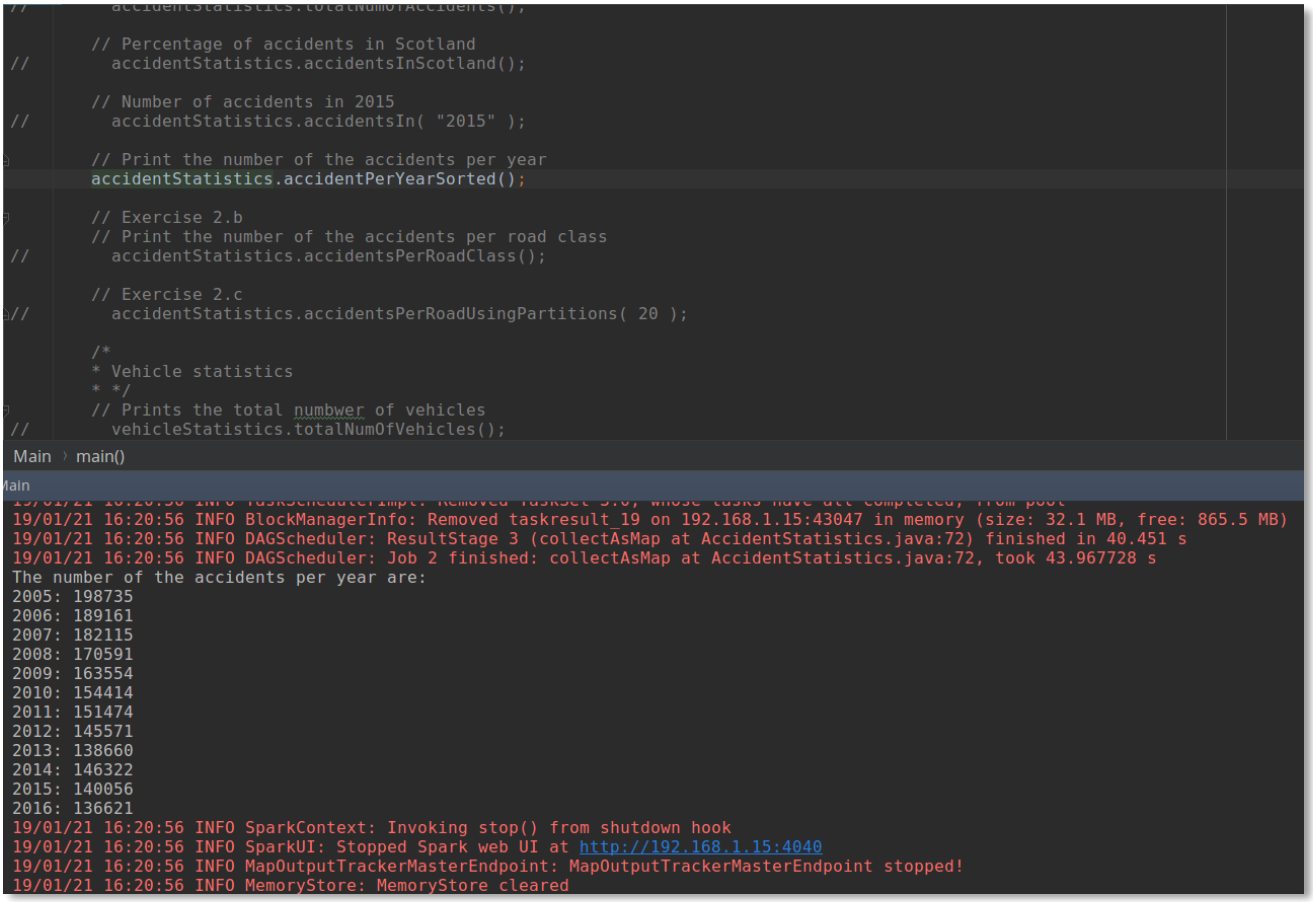
Εικόνα 2.5

3. Στην εικόνα 2.6 παρουσιάζεται το ποσοστό επι της εκατό των ατυχημάτων που έγιναν το 2015. Όπως φαίνεται, στη μέθοδο accidentsIn μπορούμε να δώσουμε ως όρισμα το έτος που θέλουμε και να εμφανιστούν τα αντίστοιχα αποτελέσματα. Για χάρη το παραδείγματος εμείς παρουσιάζουμε τα στοιχεία για το 2015 και αυτά είναι 7,3049545%.



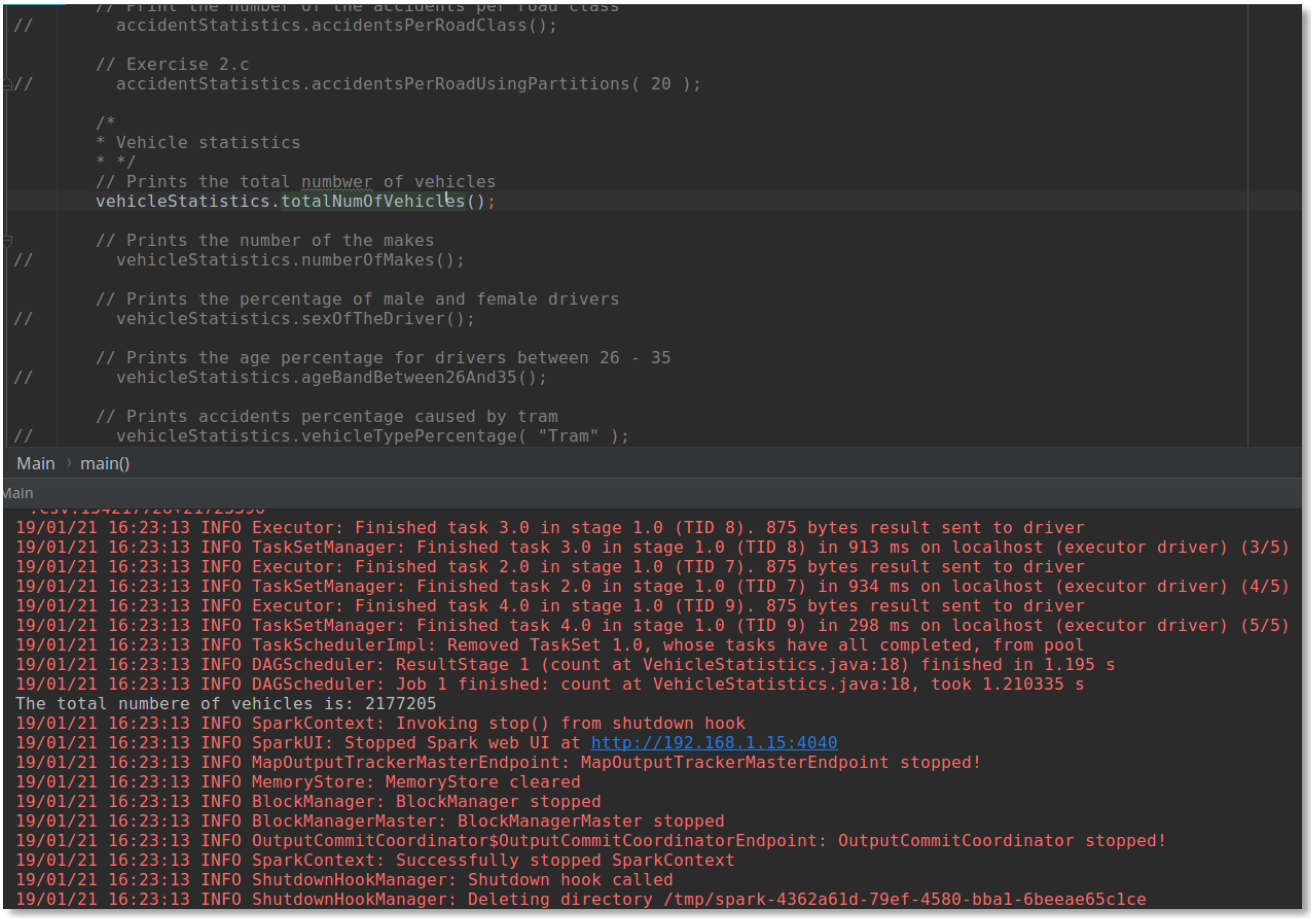
Εικόνα 2.6

4. Στην εικόνα 2.7 παρουσιάζεται το σύνολο των ατυχημάτων που συνέβησαν ανά χρόνο.



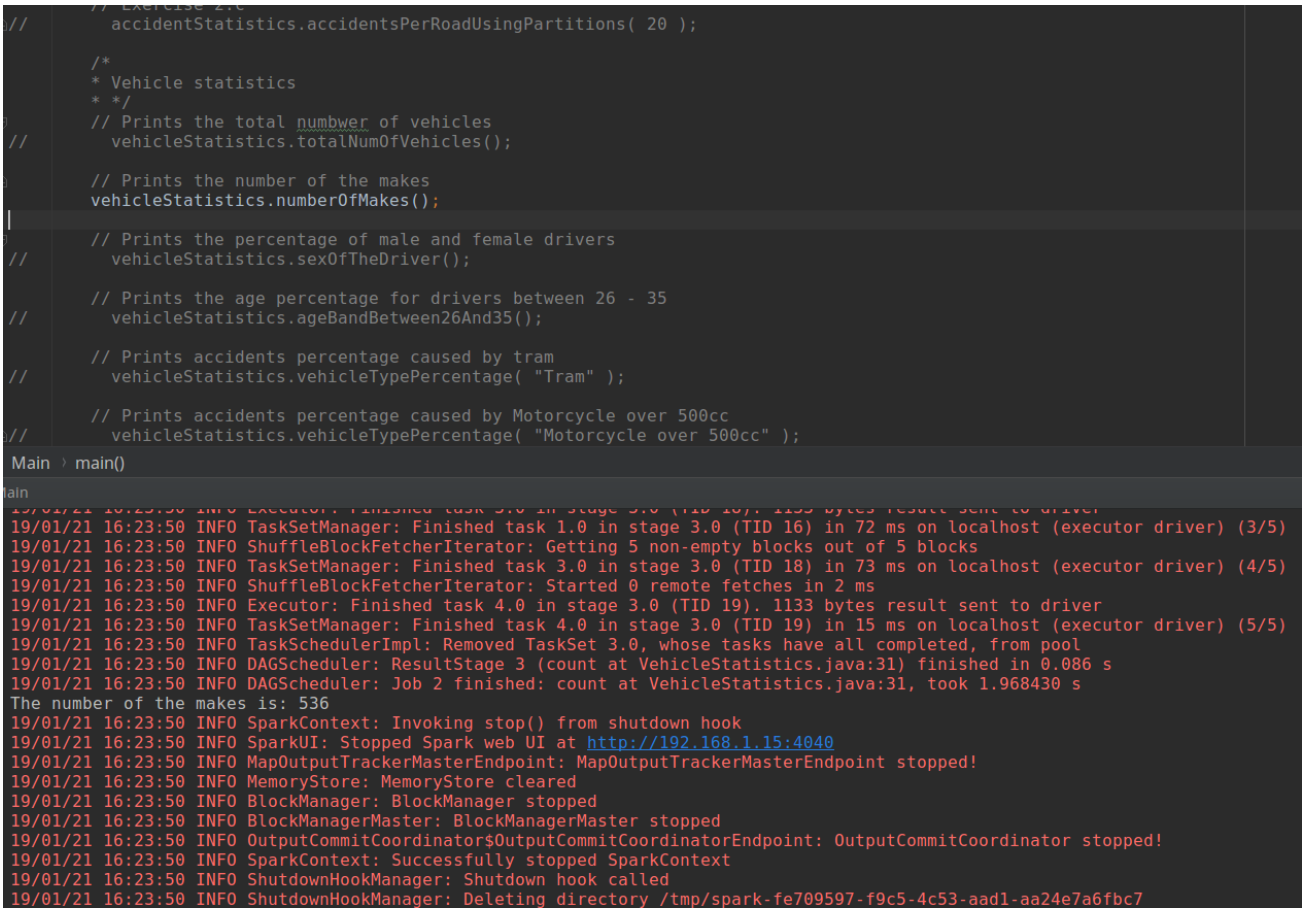
Εικόνα 2.7

5. Στην εικόνα 2.8 παρουσιάζεται ο συνολικός αριθμός των αυτοκινήτων, ο οποίος είναι 2177205.



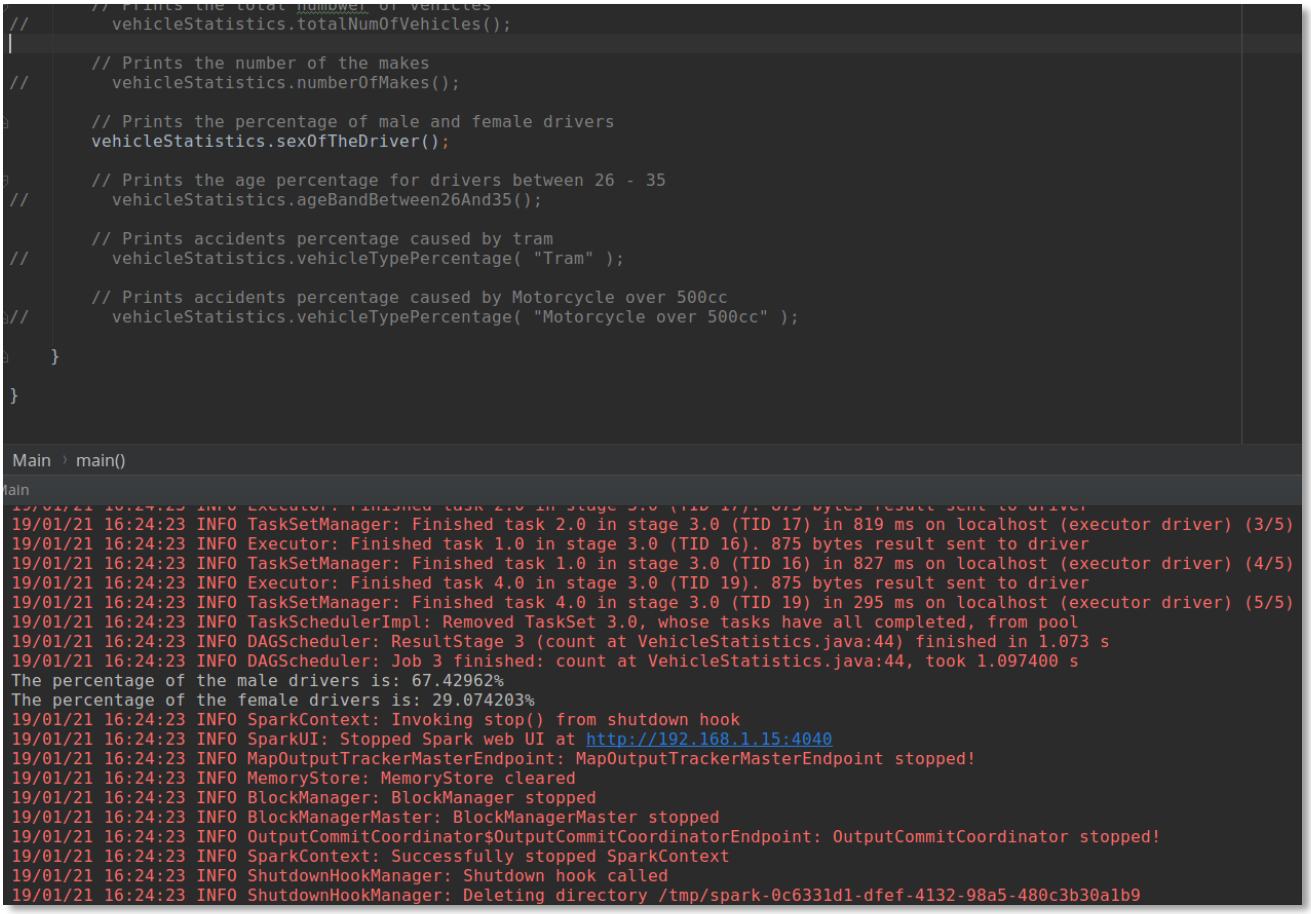
Εικόνα 2.8

6. Στην εικόνα 2.9 μπορούμε να δούμε τον πλήθος το εταιριών αυτοκινήτων. Όπως φαίνεται και στην εικόνα είναι 536.



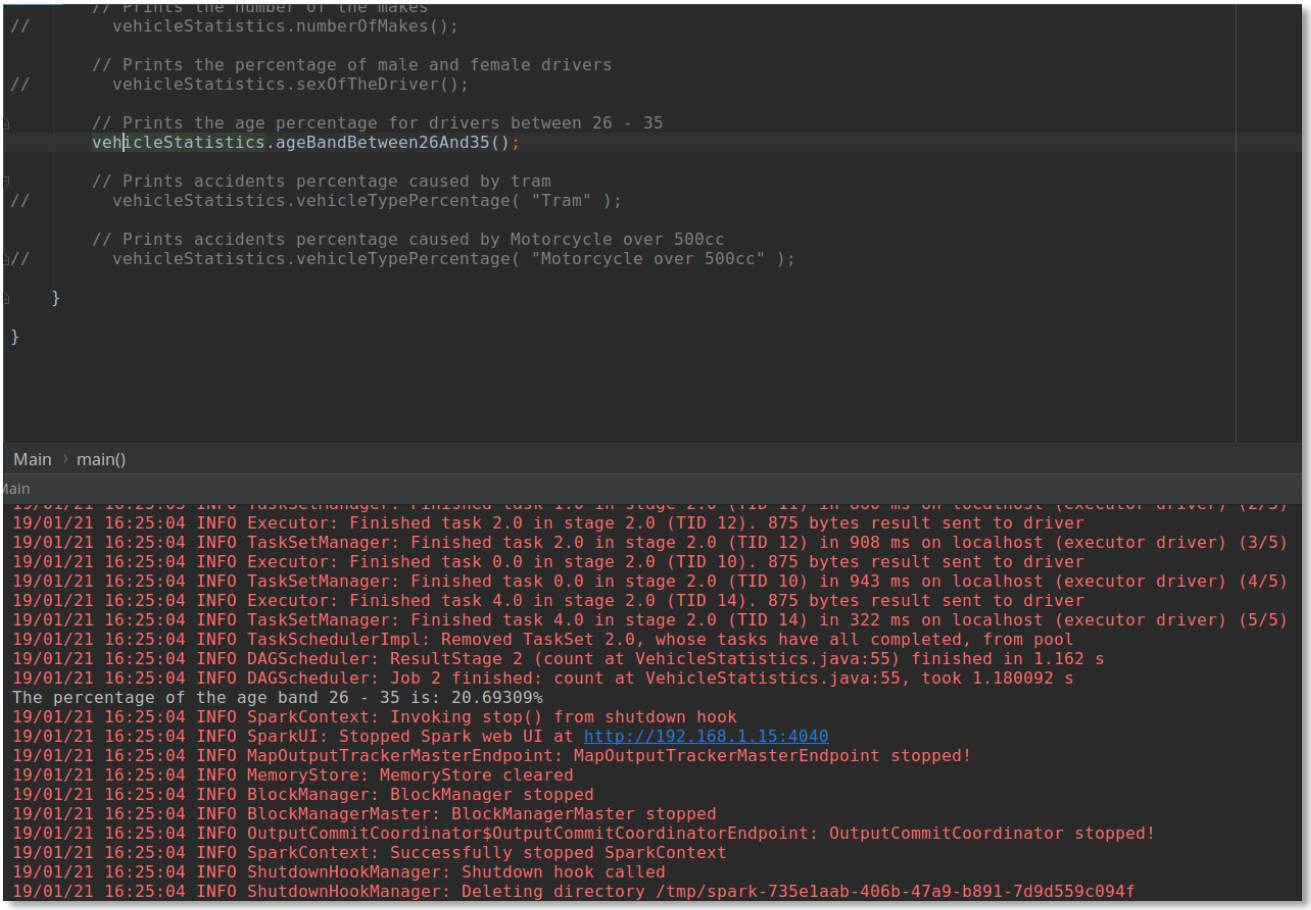
Εικόνα 2.9

7. Στην εικόνα 2.10 μπορούμε να δούμε τα ποσοστά επι της εκατό των δυο φύλων που ενεπλάκησαν σε κάποιο ατύχημα. Για τους άντρες έχουμε 67,42962% και για τις γυναίκες 29.074203%.



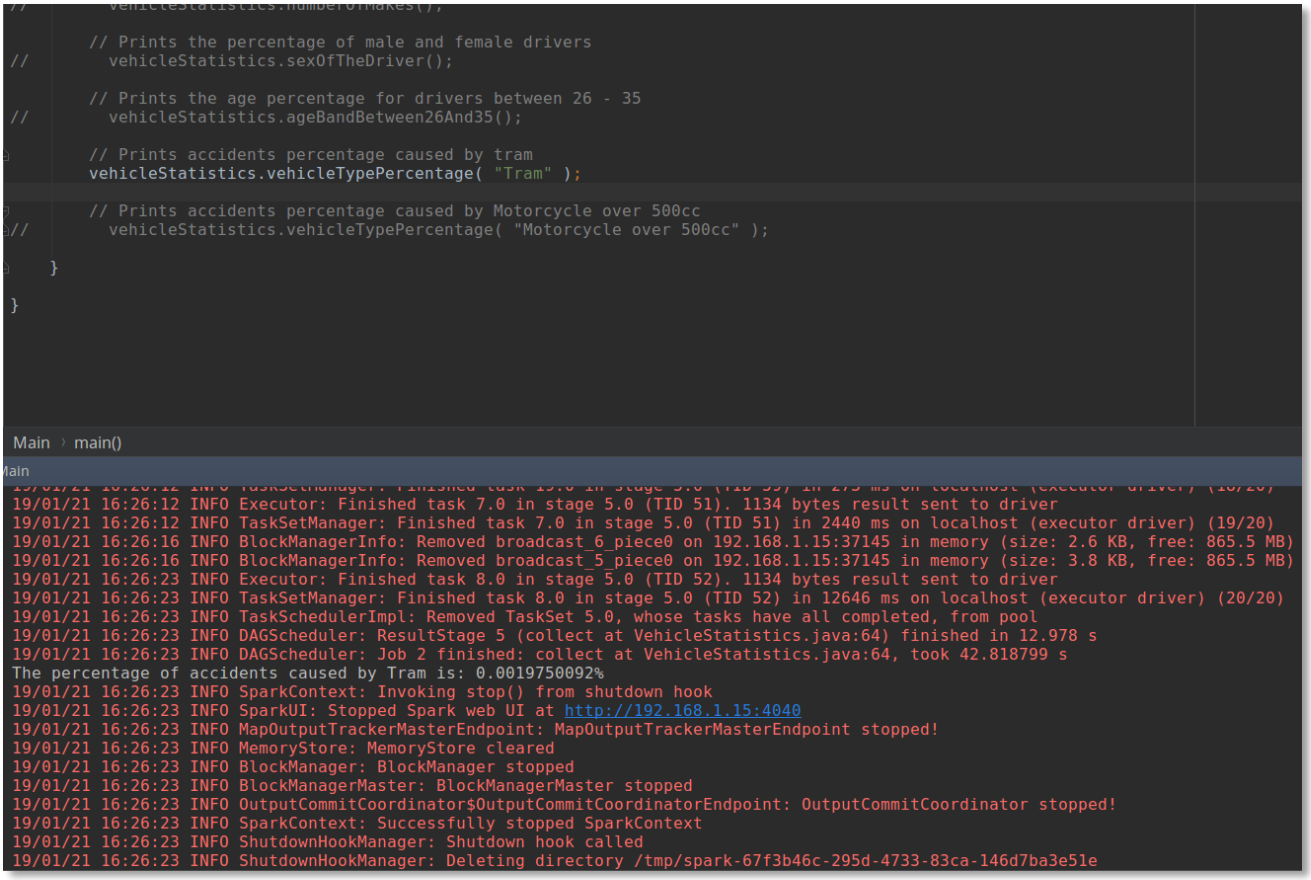
Εικόνα 2.5

8. Στην εικόνα 2.11 μπορούμε να δούμε τα ποσοστά επι της εκατό των εμπλεκομένων σε ατύχημα που ήταν μεταξύ 26 και 35 ετών.



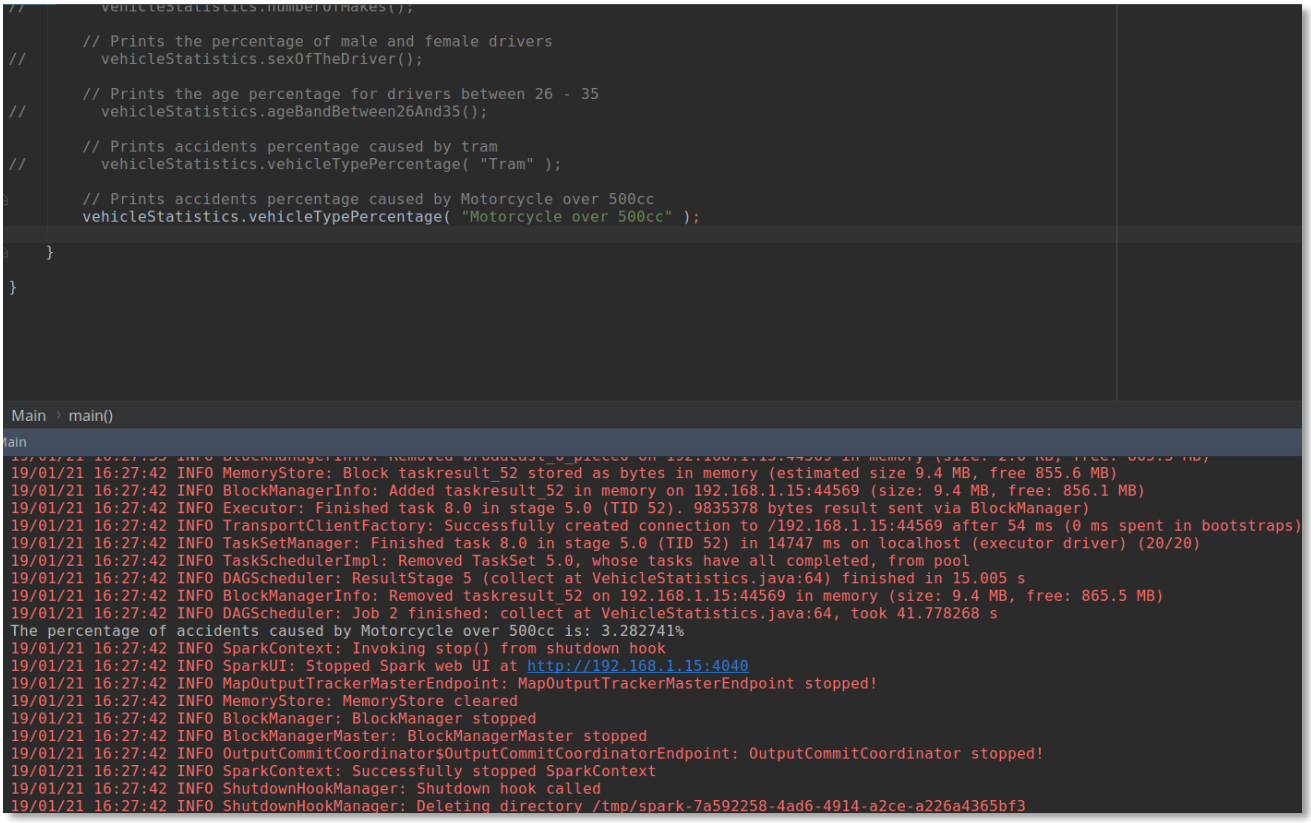
Εικόνα 2.6

9. Στην εικόνα 2.12 μπορούμε να δούμε τα ποσοστά επι της εκατό των ατυχημάτων στα οποία το όχημα ήταν τραμ.



Εικόνα 2.7

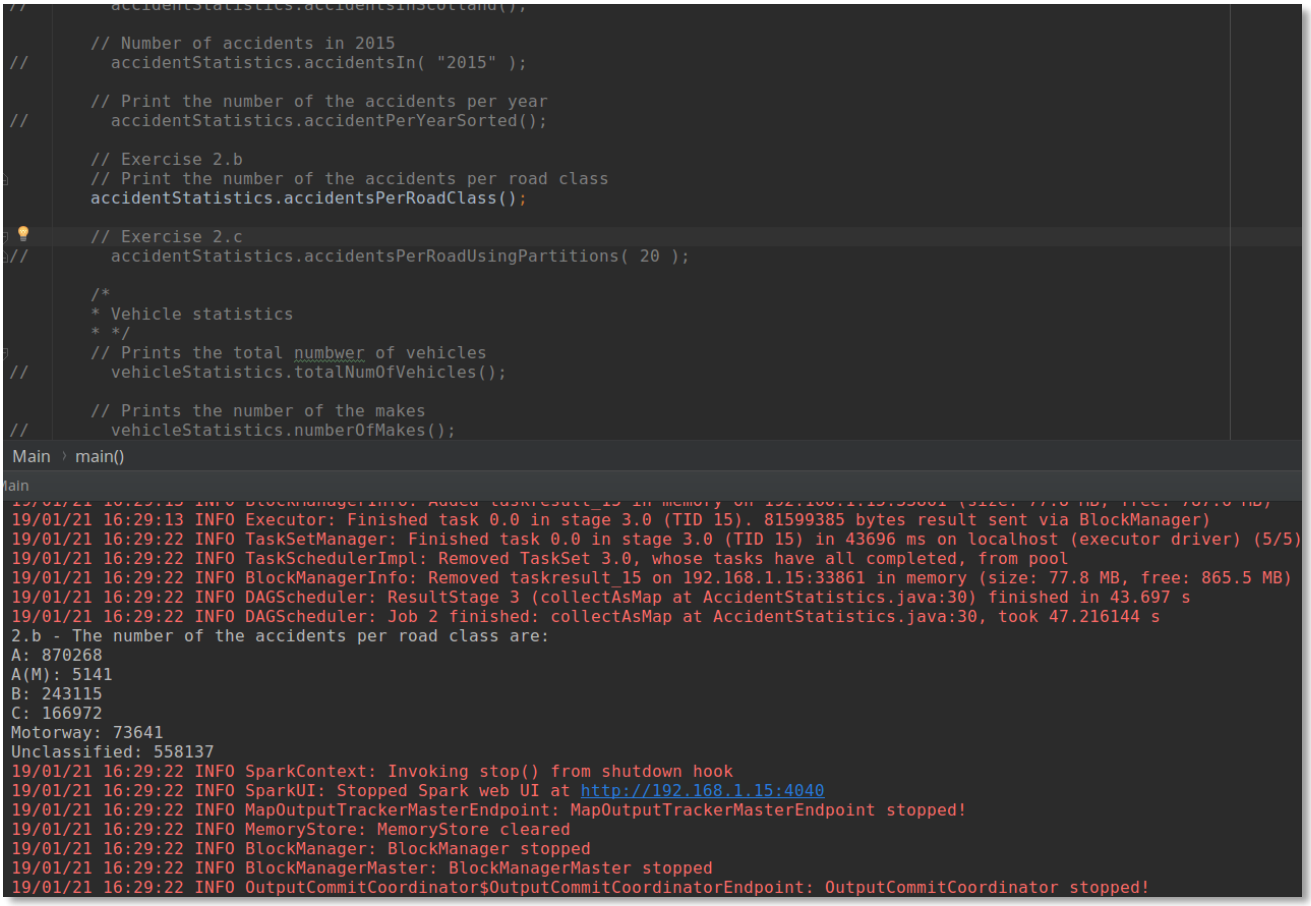
10. Στην εικόνα 2.13 μπορούμε να δούμε τα ποσοστά επι της εκατό των ατυχημάτων στα οποία το όχημα ήταν μοτοσυκλέτα άνω των 500 κυβικών.



Εικόνα 2.8

2.3 Ερώτημα Β

Το υποερωτημα που επιλέξαμε ,από το μέρος 1, ώστε να το προσαρμόσουμε στο Spark είναι το 1.a.i. Όπως φαίνεται στην εικόνα 2.14 εκτελώντας τη μέθοδο accidentsPerRoadClass παίρνουμε το αντίστοιχο αποτέλεσμα με το ερώτημα προς μετατροπή.

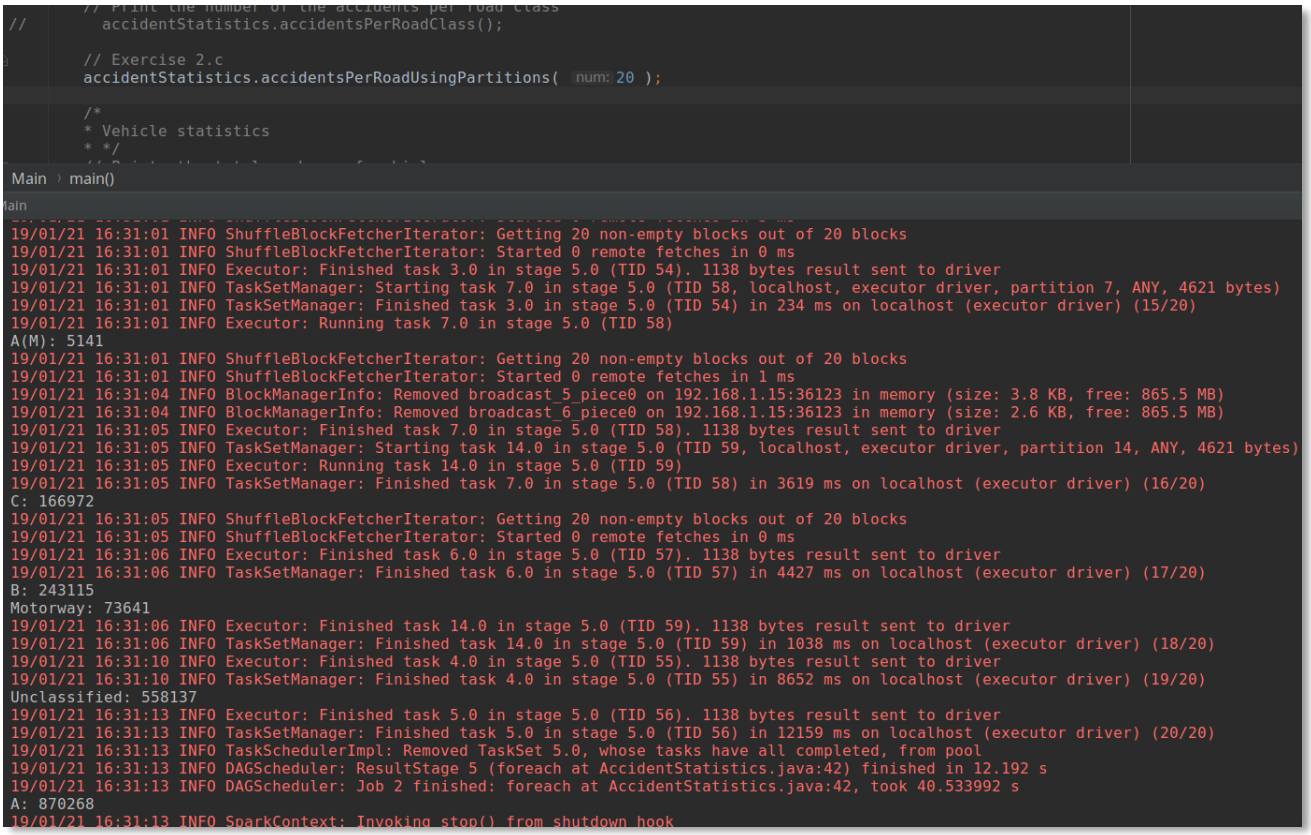


Εικόνα 2.9

2.4 Ερώτημα C

Για το τρίτο και τελευταίο ερώτημα επιλέξαμε να κάνουμε hash partitioning. Στο παράδειγμα που παρουσιάζουμε στην εικόνα 2.15 καλούμε τη μέθοδο accidentPerRoadUsingPartitions και ως όρισμα δίνουμε τον αριθμό των partition που θέλουμε. Στην περίπτωση μας, 20.

Μια παρατήρηση που μπορούμε να κάνομε είναι ότι τα αποτελέσματα εμφανίζονται ασύγχρονα σε σχέση με το ερώτημα B, δηλαδή όποτε είναι έτοιμό κάποιο αποτέλεσμά τυπώνεται στην οθόνη.



Εικόνα 2.10

2.5 Παραδοχές

Όπως είναι λογικό στα αρχεία υπήρχαν κάποια λάθη και ελλείψεις.

- Υπάρχει ένα συντακτικό λάθος στο αρχείο **db2_Vehicle_Information.csv** στη **σειρά 827652** το αν δεν διορθωθεί δεν μπορούν να φορτωθούν τα δεδομένα στη βάση δεδομένων αλλά ούτε και στο Spark.
- Το πρόβλημα της έλλειψης τιμών σε κάποια πεδία αντιμετωπίστηκε προγραμματιστικά από εμάς. Έτσι για να αποφύγουμε τα σφάλματα κατά την εκτέλεση του προγράμματος, στις τιμές που ήταν ελλείπεις, όπου ήταν απαραίτητο, δώσαμε εμείς κάποια τιμή.

Τα περισσότερο πολύπλοκα ερωτήματα στατιστικών χρειάζονται αρκετή μνήμη για να εκτελεστούν και να πράξουν αποτέλεσμα.

Σε περίπτωση που ζητηθεί, μπορούμε να παρέχουμε τα διορθωμένα αρχεία. Ήταν μεγάλα σε μέγεθος και δεν μπορούσαν να σταλούν με email.